MILITARY Intel486™ PROCESSOR FAMILY

- **IntelDX4™ Processor**
  - **— Up to 100-MHz Operation**
  - **— Speed-Multiplying Technology**
  - **— 32-Bit Architecture**
  - **— 16K-Byte On-Chip Cache**
  - **— Integrated Floating-Point Unit**
  - **— 3.3V Core Operation with 5V Tolerant I/O Buffers**
  - **— SL Technology**
  - **— Static Design**
  - **— IEEE 1149.1 Boundary Scan Compatibility**
  - **— Binary Compatible with Large Software Base**
- **IntelDX2™ Processor**
  - **— Speed-Multiplying Technology**
  - **— 32-Bit Architecture**
  - **— 8K-Byte On-Chip Cache**
  - **— Integrated Floating-Point Unit**
  - **— SL Technology**
  - **— Static Design**
  - **— IEEE 1149.1 Boundary Scan Compatibility**
  - **— Binary Compatible with Large Software Base**
- **Military Intel486™ DX Processor**
  - **— 32-Bit Architecture**
  - **— 8K-Byte On-Chip Cache**
  - **— Integrated Floating-Point Unit**
  - **— SL Technology**
  - **— Static Design**
  - **— IEEE 1149.1 Boundary Scan Compatibility**
  - **— Binary Compatible with Large Software Base**
- **All Devices Available in 168-Lead PGA Package and 196-Lead Ceramic Quad Flatpack**
- **Supported in Multiple Product Grades**

NOTE:
References to devices within this document refer to Military versions.

Military Intel486™ Processor	Product Grade			
	/B	SE1	SE2	SE3
Military Intel486™ DX Processor	✓	✓		✓
IntelDX2™ Processor		Q		✓
IntelDX4™ Processor		Q	Q	Q

Definitions:
/B = MIL-STD-883, −55°C to +125°C
SE1 = Special Environment Temperature, −55°C to +125°C
SE2 = Special Environment Temperature, −40°C to +125°C
SE3 = Special Environment Temperature, −40°C to +110°C
Q = QML qualified to MIL-STD-38535

*Other brands and names are the property of their respective owners.

July 1995 **Order Number: 271329-002**

DATA SHEET DESIGNATIONS

Intel uses various data sheet markings to designate each phase of the document as it relates to the product. The marking appears in the lower, inside corner of the data sheet. The following is the definition of these markings:

Data Sheet Marking	Description
Product Preview	Contains information on products in the design phase of development. Do not finalize a design with this information. Revised information will be published when the product becomes available.
Advance Information	Contains information on products being sampled or in the initial production phase of development.†
Preliminary	Contains preliminary information on new products in production.†
No Marking	Contains information on products in full production.†

† Specifications within these data sheets are subject to change without notice. Verify with your local Intel sales office that you have the latest datasheet before finalizing a design.

MILITARY Intel486™ PROCESSOR FAMILY

CONTENTS

CONTENTS

CONTENTS

PAGE

CONTENTS

PAGE

CONTENTS

PAGE

CONTENTS

PAGE

CONTENTS

PAGE

CONTENTS

PAGE

CONTENTS

PAGE

CONTENTS

PAGE

1.0 INTRODUCTION

The Military Intel486 processor family enables a range of low-cost, high-performance system designs for Military embedded ground, missile and Avionics applications. This family includes the IntelDX4 processor, the fastest Military Intel486 processor (up to 50% faster than an IntelDX2 processor). The IntelDX4 processor integrates a 16K unified cache and floating point hardware on-chip for improved performance. The IntelDX2 processor integrates an 8K unified cache and floating point hardware on chip. The IntelDX4 and IntelDX2 processors use Intel's speed-multiplying technology, allowing the processor to operate at frequencies higher than the external memory bus. The Military Intel486 DX processor offers the features of the IntelDX2 processors without speed-multiplying. The entire Military Intel486 processor family incorporates energy efficient ''SL Technology'' for low power computing.

SL Technology enables system designs that exceed the Environment Protection Agency's (EPA) Energy Star program guidelines, without compromising performance. It also increases system design flexibility and improves battery life in mobile applications. SLTechnology allows system designers to differentiate their power management schemes with a variety of energy-efficient or battery-life preserving features. Military Intel486 processors provide power management features that are transparent to application and operating system software. Stop Clock, Auto HALT Power Down, and Auto Idle power down allow software transparent control over processor power management. Equally important is the capability of the processor to manage system power consumption. Military Intel486 processor System Management Mode (SMM) incorporates a non-maskable System Management Interrupt (SMI#), a corresponding Resume (RSM) instruction and a new memory space for system management code. Intel's SMM ensures seamless power control of the processor core, system logic, main memory, and one or more peripheral devices, that is transparent to any application or operating system.

Military Intel486 processors are available in a full range of speeds (25 MHz to 100 MHz), packages (PGA, CQFP), and voltages (5V, 3.3V) to meet any system design requirements.

1.1 Processor Features

All of the Military Intel486 processors consist of a 32-bit integer processing unit, an on-chip cache, and a memory management unit. This ensures full binary compatibility with the 8086, 8088, 80186, 80286, Intel386™ SX, Intel386 DX, and all versions of Military Intel486 processors. All of the Military Intel486 processors offer the following features:

- *32-bit RISC integer core*—The Military Intel486 processor performs a complete set of arithmetic and logical operations on 8-, 16-, and 32-bit data types using a full-width ALU and eight general purpose registers.
- *Single Cycle Execution*—Many instructions execute in a single clock cycle.
- *Instruction Pipelining*—The fetching, decoding, address translation and execution of instructions are overlapped within the Military Intel486 processor.
- *On-Chip Floating Point Unit*—Military Intel486 processors support the 32-, 64-, and 80-bit formats specified in IEEE standard 754. The unit is binary compatible with the 8087, Intel287™, Intel387™ coprocessors.
- *On-Chip Cache with Cache Consistency Support*—An 8-Kbyte (16 Kbyte on the IntelDX4 processor) internal cache is used for both data and instructions. Cache hits provide zero wait-state access times for data within the cache. Bus activity is tracked to detect alterations in the memory represented by the internal cache. The internal cache can be invalidated or flushed so that an external cache controller can maintain cache consistency.
- *External Cache Control*—Write-back and flush controls for an external cache are provided so the processor can maintain cache consistency.
- *On-Chip Memory Management Unit*—Address management and memory space protection mechanisms maintain the integrity of memory in a multitasking and virtual memory environment. Both segmentation and paging are supported.
- *Burst Cycles*—Burst transfers allow a new double word to be read from memory on each bus clock cycle. This capability is especially useful for instruction prefetch and for filling the internal cache.

- *Write Buffers*—The processor contains four write buffers to enhance the performance of consecutive writes to memory. The processor can continue internal operations after a write to these buffers, without waiting for the write to be completed on the external bus.
- *Bus Backoff*—If another bus master needs control of the bus during a processor initiated bus cycle, the Military Intel486 processor will float its bus signals, then restart the cycle when the bus becomes available again.
- *Instruction Restart*—Programs can continue execution following an exception generated by an unsuccessful attempt to access memory. This feature is important for supporting demand-paged virtual memory applications.
- *Dynamic Bus Sizing*—External controllers can dynamically alter the effective width of the data bus. Bus widths of 8, 16, or 32 bits can be used.
- *Boundary Scan (JTAG)*—Boundary Scan provides in-circuit testing of components on printed circuit boards. The Intel Boundary Scan implementation conforms with the IEEE Standard Test Access Port and Boundary Scan Architecture.

SL Technology provides the following features:

- *Intel System Management Mode*—A unique Intel architecture operating mode provides a dedicated special purpose interrupt and address space that can be used to implement intelligent power management and other enhanced functions in a manner that is completely transparent to the operating system and applications software.
- *I/O Restart*—An I/O instruction interrupted by a System Management Interrupt (SMI#) can automatically be restarted following the execution of the RSM instruction.
- *Stop Clock*—The Military Intel486 processor has a stop clock control mechanism that provides two low-power states: a ''fast wake-up'' Stop Grant state (~20 mA–100 mA) and a ''slow wake-up'' Stop Clock state with CLK frequency at 0 MHz (100 μA–1000 μA).
- *Auto HALT Power Down*—After the execution of a HALT instruction, the Military Intel486 processor issues a normal Halt bus cycle and the clock input to the Military Intel486 processor core is automatically stopped, causing the processor to enter the Auto HALT Power Down state (~20 mA–100 mA).
- *Auto Idle Power Down* —This function allows the processor to reduce the core frequency to the bus frequency when both the core and bus are idle. Auto Idle Power Down is software transparent and does not affect processor performance. Auto Idle Power Down provides an average power savings of 10% and is only applicable to clock multiplied processors.

1.2 Military Intel486™ Processor Product Family

Table 1-1 shows the Military Intel486 processors available by Maximum Frequency and Package.

Table 1-1. Product Options

Military Intel486™ Processor	Processor Frequency (MHz)						Package Type	
	25	33	50	66	75	100	168-Pin PGA	196-Lead CQFP
Military Intel486™ DX Processor	✓	✓					✓	✓
IntelDX2™ Processor			✓	✓			✓	✓
IntelDX4™ Processor					✓	✓	✓	✓

2.0 HOW TO USE THIS DOCUMENT

2.1 Introduction

This data sheet is a compilation of previously published individual data sheets for the Military Intel486 DX, IntelDX2 and IntelDX4 processors. This data sheet encompasses the entire current Military Intel486 processor family.

This data sheet describes the Military Intel486 processor architecture, features and technical details. Unless otherwise stated, any description for the Military Intel486 processor listed in this data sheet applies to all Military Intel486 processors. Where architectural or other differences do occur (for example, the IntelDX4 processor has a 16-Kbyte on-chip cache, all other Military Intel486 processors have an 8-Kbyte on-chip cache), these differences are described in separate sections. Section 2.2 provides a brief section description, highlighting the specific sections that contain processor-unique information.

It is important to note that all Military Intel486 DX, IntelDX2, and IntelDX4 processors have an on-chip floating point unit.

Boundary Scan (JTAG) testability features, capability and associated test signals (TCK, TMS, TDI, and TDO) are standard on all Military Intel486 processors.

2.2 Section Contents and Processor Specific Information

The following is a brief description of the contents of each section:

Section 1: ''Introduction.'' This section is an overview of the current Military Intel486 processor family, product features and highlights. This section also lists product frequency, voltage and package offerings.

Section 2: ''How to Use This Document.'' This section presents information to aid in the use of this data sheet.

Section 3: ''Pin Description.'' This section contains all of the pin configurations for the various package options (168-Pin PGA and 196-Lead CQFP), package diagrams, pin assignment tables and pin assignment differences for the various processors within a package class.

This section also provides a quick pin reference table that lists pin signals for the Military Intel486 processor family. The table, whenever necessary, has sections applicable to each current Military Intel486 processor family member.

Section 4: ''Architectural Overview.'' This section describes the Military Intel486 processor architecture, including the register and instruction sets, memory organization, data types and formats, and interrupts for all Military Intel486 processors.

The architectural overview describes the 32-bit RISC integer core of the Military Intel486 processor. The on-chip floating point unit for the Military Intel486 DX IntelDX2 and IntelDX4 processors is included in this section.

Section 5: ''Real Mode Architecture.'' This section describes the Military Intel486 processor real-mode architecture, including memory addressing, reserved locations, interrupts, and Shutdown and HALT. This section applies to all Military Intel486 processors.

Section 6: ''Protected Mode Architecture.'' This section describes the Military Intel486 protected-mode architecture, including addressing mechanism, segmentation, protection, paging and virtual 8086 environment. This section applies to all Military Intel486 processors.

Section 7: ''On-Chip Cache.'' This section describes the on-chip cache of the Military Intel486 processors. Specific information on size, features, modes, and configurations is described. The differences between the IntelDX4 processor on-chip cache (16-KByte) and other members of the Military Intel486 processor family on chip cache (8-KByte) are detailed.

Section 8: ''System Management Mode (SMM) Architectures.'' This section describes the System Management Mode architecture of the Military Intel486 processors, including system management mode interrupt processing and programming mode.

Section 9: ''Hardware Interface.'' This section describes the hardware interface of the current Military Intel486 processor family, including signal descriptions, interrupt interfaces, write buffers, reset and initialization, and clock control.

The IntelDX4 processor speed multiplying options are detailed in this section. Reset and initialization, as it applies to all of the Military Intel486 processor family, is also documented here.

Use and operation of the Stop Clock, Auto HALT Power Down and other power-saving SL Technology features are described.

Section 10: ''Bus Operation.'' This section describes the Military Intel486 processor bus operation, including the data transfer mechanism and bus functional description.

Section 11: ''Testability.'' This section describes the testability of the Military Intel486 processors, including the built-in self test (BIST), on-chip cache testing, translation lookaside buffer (TLB) testing, tri-state output test mode, and boundary scan (JTAG).

Section 12: ''Debugging Support.'' This section describes the Military Intel486 processor debugging support, including the breakpoint instruction, single-step trap and debug registers. This section applies to all Military Intel486 processors.

Section 13: ''Instruction Set Summary.'' This section provides clock count and instruction encoding summaries for all the Military Intel486 processors.

Section 14: ''Differences between Military Intel486 Processors and Intel386™ Processors.'' This section lists the differences between the Military Intel486 processor family and the Intel386 processor family. Also described and documented are differences between the Intel386 with an Intel387™ math coprocessors and the Military Intel486 processors with on-chip floating point units. This section applies to all Military Intel486 processors.

Section 15: ''Electrical Data.'' This section lists the AC and DC specifications for all Military Intel486 processors. Processor specific information is listed in both common and separate tables and sections as appropriate.

Section 16: ''Mechanical Data.'' This section lists the mechanical and thermal data, including the package specifications (PGA and CQFP) for all Military Intel486 processors. Processor specific information is listed in both common and separate tables and sections as appropriate.

Appendix A: ''Features Determination.'' This section documents the CPUID function to determine the Military Intel486 processor family identification and processor specific information. This section applies to all Military Intel486 processors.

Appendix B: ''IBIS Models.'' This section provides a detailed sample listing of the types of I/O buffer modeling information available for the Military Intel486 processor family. This section applies to all Military Intel486 processors.

Appendix C: ''BSDL Listing.'' This section provides a sample listing of a BSDL file for the Military Intel486 processor family. This section applies to all Military Intel486 processors.

Appendix D: ''System Design Notes.'' This section provides design notes applicable to the use of System Management Mode and SMM routines with the Military Intel486 processor.

2.3 Documents Replaced by This Data Sheet

This Data Sheet contains all of the latest information for the Military Intel486 processor family and replaces the following documentation:

Military i486 ™ *DX Microprocessor Datasheet*, Order Number 271136

Military Intel486 ™ *DX2 Microprocessor Datasheet*, Order Number 271280

3.0 PIN DESCRIPTION

3.1 Pin Assignments

The following figures show the pin assignments of each package type for the Military Intel486 processor product family. Tables are provided showing the pin differences between the existing Military Intel486 processor products and the Military Intel486 processor products.

168-Pin PGA—Pin Grid Array

- Package Diagram
- Pin Assignment Difference Table
- Pin Cross Reference by Pin Name

196-Lead CQFP—Quad Flat Pack

- Package Diagram
- Pin Assignment Difference Table
- Pin Assignment Table in numerical order

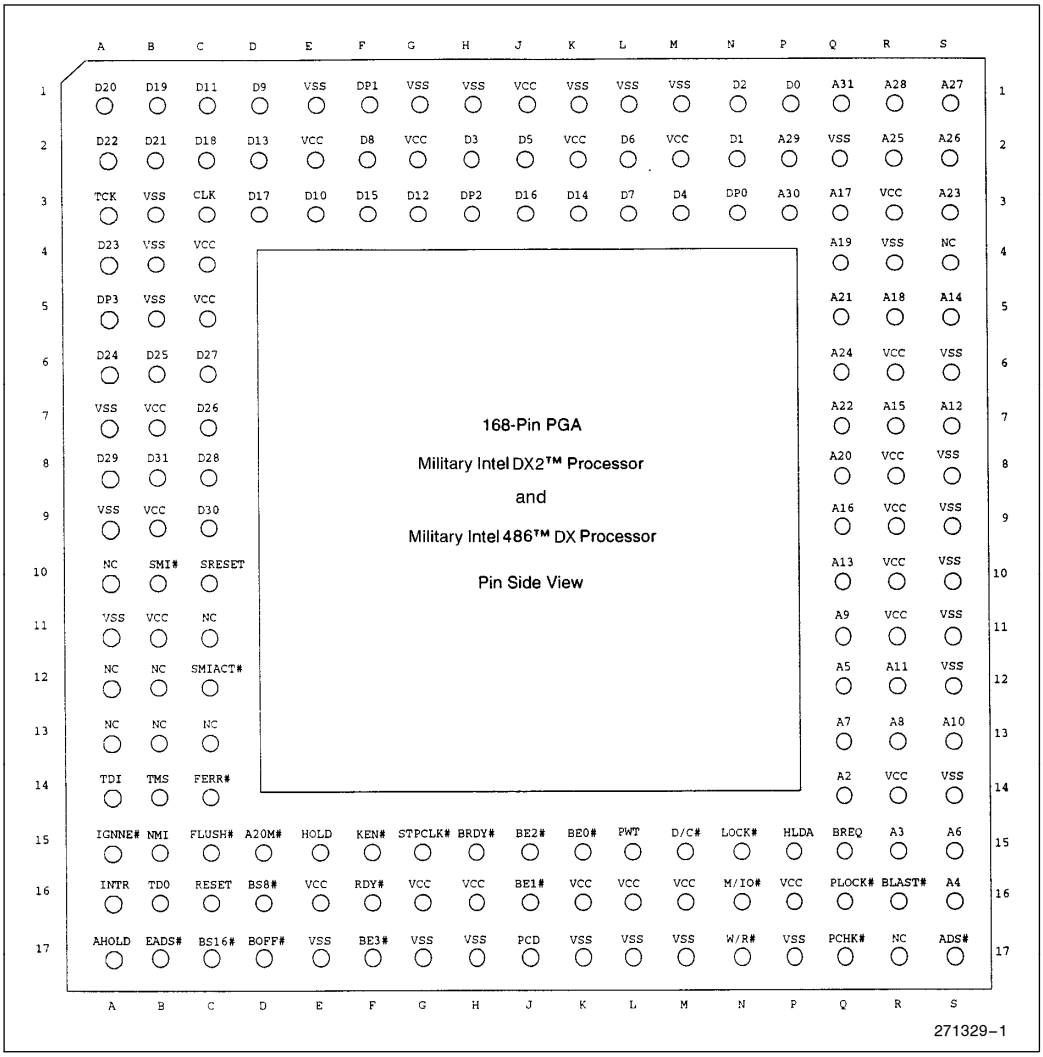


Figure 3-1. Package Diagram for 168-Pin PGA Package of the Military IntelDX2™ Processor and the Military Intel486™ DX Processor

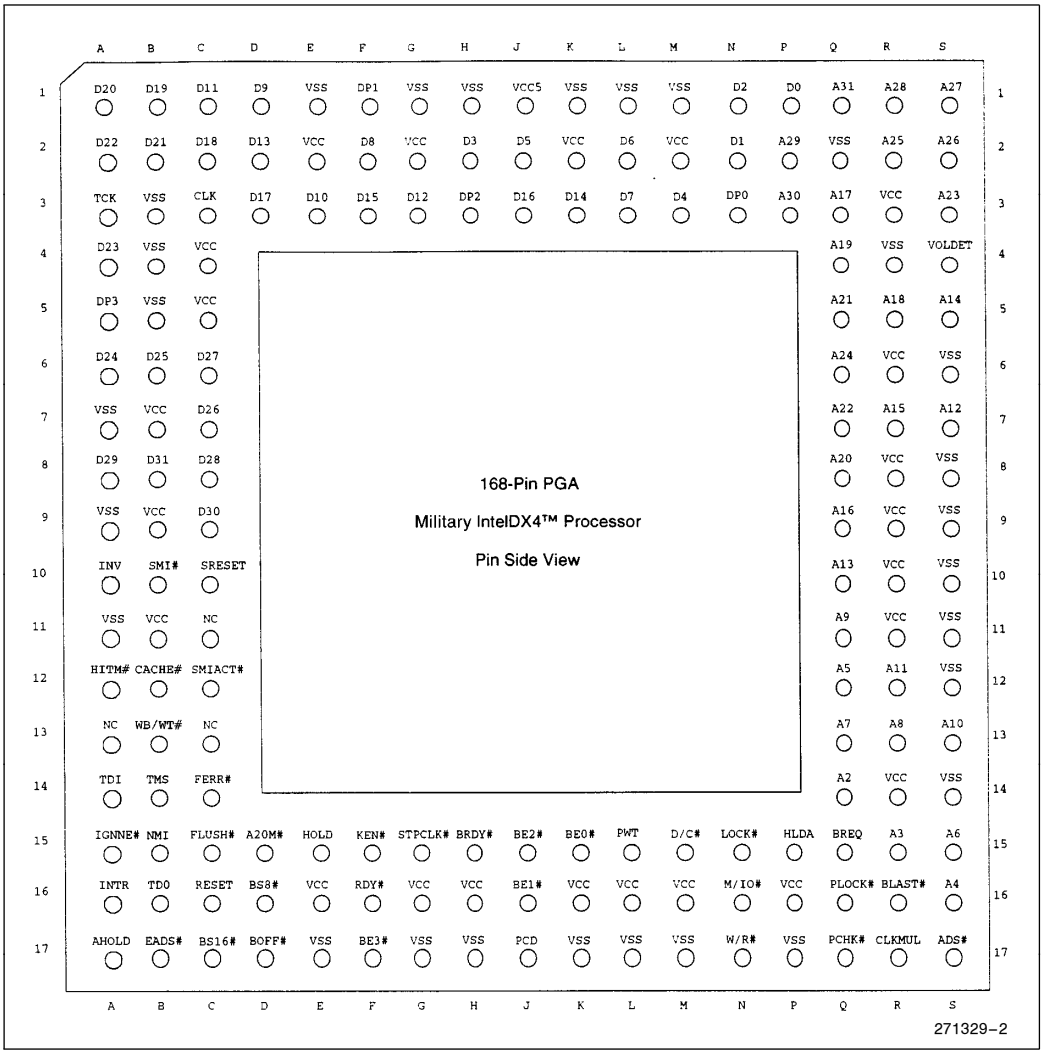


Figure 3-2. 168-Pin PGA Pinout Diagram (Pin Side) for the Military IntelDX4™ Processor

Table 3-1. Pinout Differences for 168-Pin PGA Package

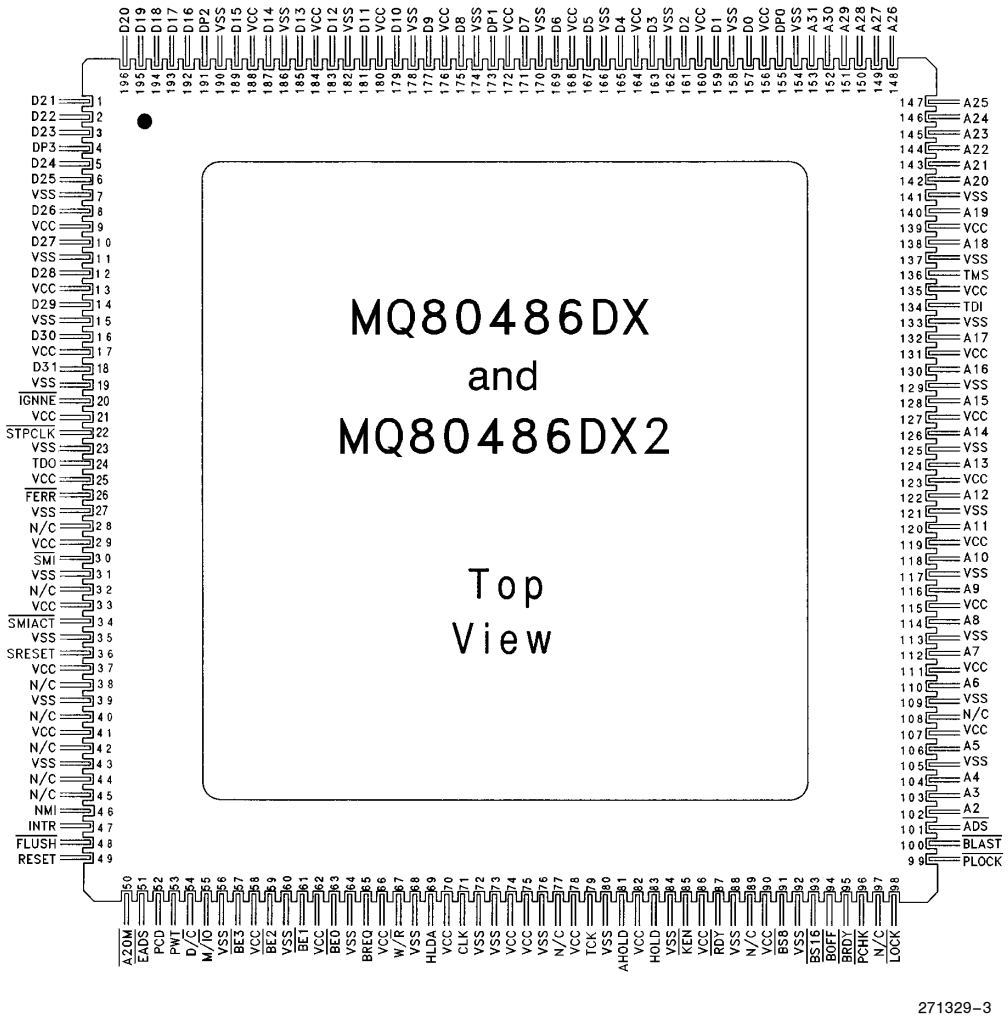
Pin	Previous Military Intel486 DX Processor(4)	Military Intel486 DX Processor	Previous IntelDX2™ Processor(4)	IntelDX2 Processor	IntelDX4™ Processor
A3(3)	NC	TCK	TCK	TCK	TCK
A14(3)	NC	TDI	TDI	TDI	TDI
B10(3)	NC	SMI#	NC	SMI#	SMI#
B14(3)	NC	TMS	TMS	TMS	TMS
B16(3)	NC	TDO	TDO	TDO	TDO
C10(3)	NC	SRESET	NC	SRESET	SRESET
C12	NC	SMIACT#	NC	SMIACT#	SMIACT#
G15(3)	NC	STPCLK#	NC	STPCLK#	STPCLK#
J1	V_{CC}	V_{CC}	V_{CC}	V_{CC}	V_{CC5}(2)
R17	NC	NC	NC	NC	CLKMUL
S4	NC	NC	NC	NC	VOLDET
A10	NC	NC	NC	NC	INV
A12	NC	NC	NC	NC	HITM#
B12	NC	NC	NC	NC	CACHE#
B13	NC	NC	NC	NC	WB/WT#

NOTES:
1. NC. Do Not Connect. These pins should always remain unconnected. Connection of NC pins to V_{CC} or V_{SS} or to any other signal can result in component malfunction or incompatibility with future steppings of the Military Intel486 processors.
2. This pin location is for the V_{CC5} pin on the IntelDX4 processor. For compatibility with 3.3V processors that have 5V safe input buffers (i.e., IntelDX4 processors), this pin should be connected to a V_{CC} trace, not to the V_{CC} plane. See section 3.2, ''Quick Pin Reference,'' for a description of the V_{CC5} pin on the IntelDX4 processor.
3. These pins were No Connects on previous Military Intel486 DX and IntelDX2 processors. For compatibility with old designs, they can still be left unconnected.
4. Previous versions of the Military Intel486 processor family do not implement SL Technology and are not described in this data sheet.

Table 3-2. Pin Cross Reference for 168-Pin PGA Package of the IntelDX2™ Processor, Intel486™ DX Processor and IntelDX4™ Processor

Address	Data	Control	NC (2)	V_{CC}	V_{SS}
A2 .....Q14	D0.....P1	A20M# D15	A10	B7	A7
A3 .....R15	D1 .....N2	ADS# S17	A12	B9	A9
A4.....S16	D2 .....N1	AHOLD A17	A13	B11	A11
A5 .....Q12	D3 .....H2	BE0# K15	B12	C4	B3
A6.....S15	D4 .....M3	BE1# J16	B13	C5	B4
A7 .....Q13	D5 .....J2	BE2# J15	C11	E2	B5
A8 .....R13	D6 .....L2	BE3# F17	C13	E16	E1
A9 .....Q11	D7 .....L3	BLAST# R16	R17	G2	E17
A10 ....S13	D8 .....F2	BOFF# D17	S4	G16	G1
A11 ....R12	D9 .....D1	BRDY# H15		H16	G17
A12.....S7	D10 .....E3	BREQ Q15		J1	H17
A13 ....Q10	D11 .....C1	BS8# D16		K2	H1
A14.....S5	D12 .....G3	BS16# C17		K16	K1
A15 .....R7	D13 .....D2	CLK C3		L16	K17
A16 .....Q9	D14 .....K3	D/C# M15		M2	L1
A17 .....Q3	D15.....F3	DP0 .....N3		M16	L17
A18 .....R5	D16.....J3	DP1 .....F1		P16	M1
A19 .....Q4	D17 .....D3	DP2 .....H3		R3	M17
A20 .....Q8	D18 .....C2	DP3 .....A5		R6	P17
A21 .....Q5	D19 .....B1	EADS# B17		R8	Q2
A22 .....Q7	D20 .....A1	FERR# C14		R9	R4
A23.....S3	D21 .....B2	FLUSH# C15		R10	S6
A24 .....Q6	D22 .....A2	HLDA P15		R11	S8
A25 .....R2	D23 .....A4	HOLD E15		R14	S9
A26.....S2	D24 .....A6	IGNNE# A15		V_{CC5}(1)	S10
A27.....S1	D25 .....B6	INTR A16			S11
A28 .....R1	D26 .....C7	KEN# F15		J1	S12
A29.....P2	D27 .....C6	LOCK# N15			S14
A30.....P3	D28 .....C8	M/IO# N16			
A31 .....Q1	D29 .....A8	NMI B15			
	D30 .....C9	PCD J17			
	D31 .....B8	PCHK# Q17			
		PWT L15			
		PLOCK Q16			
		RDY# F16			
		RESET C16			
		SMI# B10			
		SMIACT# ...C12			
		W/R# N17			
		STPCLK# ..G15			
		SRESET C10			
		TCK A3			
		TDI A14			
		TDO B16			
		TMS B14			
		VOLDET(3) ...S4			
		CLKMUL(3) ..R17			
		CACHE# ...B12			
		HITM# A12			
		INV A10			
		WB/WT# ...B13			

NOTES:
1. V_{CC5} is for the IntelDX4 processor only.
2. NC. Do Not Connect. These pins should always remain unconnected. Connection of NC pins to V_{CC} or V_{SS} or to any other signal can result in component malfunction or incompatibility with future steppings of the Military Intel486 processors.
3. Present only on the IntelDX4 processor.



NOTE:
SL Enhanced features not available on MQ80486DX. Pins 22, 30, 34, and 36 are "No Connects" on MQ80486DX.

Figure 3-3. 196-Pin Ceramic Quad Flatpack (CQFP) Pin Configuration—View from Lid Side

Military Intel486™ DX and Intel DX2™ CQFP Pin Cross Reference

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	D21	35	V_{SS}	69	HLDA	103	A3	137	V_{SS}	171	D7
2	D22	36	SRESET	70	V_{CC}	104	A4	138	A18	172	V_{CC}
3	D23	37	V_{CC}	71	CLK	105	V_{SS}	139	V_{CC}	173	DP1
4	DP3	38	N/C	72	V_{SS}	106	A5	140	A19	174	V_{SS}
5	D24	39	V_{SS}	73	V_{SS}	107	V_{CC}	141	V_{SS}	175	D8
6	D25	40	N/C	74	V_{CC}	108	N/C	142	A20	176	V_{CC}
7	V_{SS}	41	V_{CC}	75	V_{CC}	109	V_{SS}	143	A21	177	D9
8	D26	42	N/C	76	V_{SS}	110	A6	144	A22	178	V_{SS}
9	V_{CC}	43	V_{SS}	77	N/C	111	V_{CC}	145	A23	179	D10
10	D27	44	N/C	78	V_{CC}	112	A7	146	A24	180	V_{CC}
11	V_{SS}	45	N/C	79	TCK	113	V_{SS}	147	A25	181	D11
12	D28	46	NMI	80	V_{SS}	114	A8	148	A26	182	V_{SS}
13	V_{CC}	47	INTR	81	AHOLD	115	V_{CC}	149	A27	183	D12
14	D29	48	$\overline{FLUSH}$	82	V_{CC}	116	A9	150	A28	184	V_{CC}
15	V_{SS}	49	RESET	83	HOLD	117	V_{SS}	151	A29	185	D13
16	D30	50	$\overline{A20M}$	84	V_{SS}	118	A10	152	A30	186	V_{SS}
17	V_{CC}	51	$\overline{EADS}$	85	$\overline{KEN}$	119	V_{CC}	153	A31	187	D14
18	D31	52	PCD	86	V_{CC}	120	A11	154	V_{SS}	188	V_{CC}
19	V_{SS}	53	PWT	87	$\overline{RDY}$	121	V_{SS}	155	DP0	189	D15
20	$\overline{IGNNE}$	54	D/$\overline{C}$	88	V_{SS}	122	A12	156	V_{CC}	190	V_{SS}
21	V_{CC}	55	M/$\overline{IO}$	89	N/C	123	V_{CC}	157	D0	191	DP2
22	$\overline{STPCLK}$	56	V_{SS}	90	V_{CC}	124	A13	158	V_{SS}	192	D16
23	V_{SS}	57	$\overline{BE3}$	91	$\overline{BS8}$	125	V_{SS}	159	D1	193	D17
24	TDO	58	V_{CC}	92	V_{SS}	126	A14	160	V_{CC}	194	D18
25	V_{CC}	59	$\overline{BE2}$	93	$\overline{BS16}$	127	V_{CC}	161	D2	195	D19
26	$\overline{FERR}$	60	V_{SS}	94	$\overline{BOFF}$	128	A15	162	V_{SS}	196	D20
27	V_{SS}	61	$\overline{BE1}$	95	$\overline{BRDY}$	129	V_{SS}	163	D3		
28	N/C	62	V_{CC}	96	$\overline{PCHK}$	130	A16	164	V_{CC}		
29	V_{CC}	63	$\overline{BE0}$	97	N/C	131	V_{CC}	165	D4		
30	$\overline{SMI}$	64	V_{SS}	98	$\overline{LOCK}$	132	A17	166	V_{SS}		
31	V_{SS}	65	BREQ	99	$\overline{PLOCK}$	133	V_{SS}	167	D5		
32	N/C	66	V_{CC}	100	$\overline{BLAST}$	134	TDI	168	V_{CC}		
33	V_{CC}	67	W/$\overline{R}$	101	$\overline{ADS}$	135	V_{CC}	169	D6		
34	$\overline{SMIACT}$	68	V_{SS}	102	A2	136	TMS	170	V_{SS}		

NOTES:
No Connect (N/C) pins must not be connected.
SL Enhanced features not available on MQ80486DX. Pins 22, 30, 34, and 36 are ''No Connects'' on MQ80486DX.

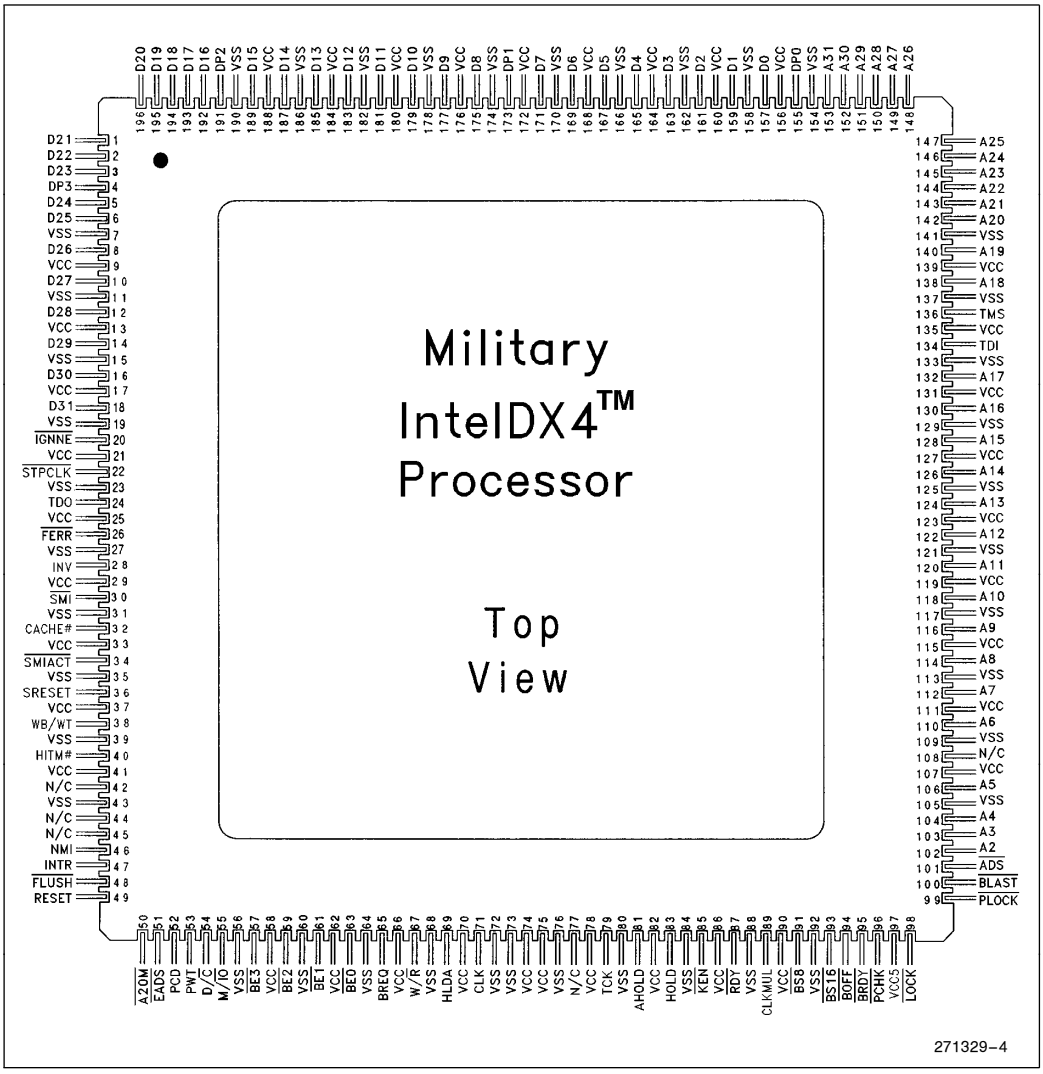


Figure 3-4. 196-Pin Ceramic Quad Flatpack (CQFP) Pin Configuration—View from Lid Side

Military IntelDX4™ CQFP Pin Cross Reference

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	D21	35	V_{SS}	69	HLDA	103	A3	137	V_{SS}	171	D7
2	D22	36	SRESET	70	V_{CC}	104	A4	138	A18	172	V_{CC}
3	D23	37	V_{CC}	71	CLK	105	V_{SS}	139	V_{CC}	173	DP1
4	DP3	38	WB/WT#	72	V_{SS}	106	A5	140	A19	174	V_{SS}
5	D24	39	V_{SS}	73	V_{SS}	107	V_{CC}	141	V_{SS}	175	D8
6	D25	40	HITM#	74	V_{CC}	108	N/C	142	A20	176	V_{CC}
7	V_{SS}	41	V_{CC}	75	V_{CC}	109	V_{SS}	143	A21	177	D9
8	D26	42	N/C	76	V_{SS}	110	A6	144	A22	178	V_{SS}
9	V_{CC}	43	V_{SS}	77	N/C	111	V_{CC}	145	A23	179	D10
10	D27	44	N/C	78	V_{CC}	112	A7	146	A24	180	V_{CC}
11	V_{SS}	45	N/C	79	TCK	113	V_{SS}	147	A25	181	D11
12	D28	46	NMI	80	V_{SS}	114	A8	148	A26	182	V_{SS}
13	V_{CC}	47	INTR	81	AHOLD	115	V_{CC}	149	A27	183	D12
14	D29	48	$\overline{FLUSH}$	82	V_{CC}	116	A9	150	A28	184	V_{CC}
15	V_{SS}	49	RESET	83	HOLD	117	V_{SS}	151	A29	185	D13
16	D30	50	$\overline{A20M}$	84	V_{SS}	118	A10	152	A30	186	V_{SS}
17	V_{CC}	51	$\overline{EADS}$	85	$\overline{KEN}$	119	V_{CC}	153	A31	187	D14
18	D31	52	PCD	86	V_{CC}	120	A11	154	V_{SS}	188	V_{CC}
19	V_{SS}	53	PWT	87	$\overline{RDY}$	121	V_{SS}	155	DP0	189	D15
20	$\overline{IGNNE}$	54	D/$\overline{C}$	88	V_{SS}	122	A12	156	V_{CC}	190	V_{SS}
21	V_{CC}	55	M/$\overline{IO}$	89	CLKMUL	123	V_{CC}	157	D0	191	DP2
22	$\overline{STPCLK}$	56	V_{SS}	90	V_{CC}	124	A13	158	V_{SS}	192	D16
23	V_{SS}	57	$\overline{BE3}$	91	$\overline{BS8}$	125	V_{SS}	159	D1	193	D17
24	TDO	58	V_{CC}	92	V_{SS}	126	A14	160	V_{CC}	194	D18
25	V_{CC}	59	$\overline{BE2}$	93	$\overline{BS16}$	127	V_{CC}	161	D2	195	D19
26	$\overline{FERR}$	60	V_{SS}	94	$\overline{BOFF}$	128	A15	162	V_{SS}	196	D20
27	V_{SS}	61	$\overline{BE1}$	95	$\overline{BRDY}$	129	V_{SS}	163	D3		
28	INV	62	V_{CC}	96	$\overline{PCHK}$	130	A16	164	V_{CC}		
29	V_{CC}	63	$\overline{BE0}$	97	V_{CC5}	131	V_{CC}	165	D4		
30	$\overline{SMI}$	64	V_{SS}	98	$\overline{LOCK}$	132	A17	166	V_{SS}		
31	V_{SS}	65	BREQ	99	$\overline{PLOCK}$	133	V_{SS}	167	D5		
32	CACHE#	66	V_{CC}	100	$\overline{BLAST}$	134	TDI	168	V_{CC}		
33	V_{CC}	67	W/$\overline{R}$	101	$\overline{ADS}$	135	V_{CC}	169	D6		
34	$\overline{SMIACT}$	68	V_{SS}	102	A2	136	TMS	170	V_{SS}		

NOTE:
No Connect (N/C) pins must not be connected.

3.2 Quick Pin Reference

The following is a brief pin description. For detailed signal descriptions refer to section 9.2, ''Signal Description.''

Table 3-3. Military Intel486™ Processor Pin Descriptions

Symbol	Type	Name and Function
CLK	I	**CLocK** provides the fundamental timing and the internal operating frequency for the Military Intel486 processor. All external timing parameters are specified with respect to the rising edge of CLK.
ADDRESS BUS		
A31–A4 **A2–A3**	I/O O	The **Address Lines.** A31–A2, together with the byte enables signals. BE0#–BE3#, define the physical area of memory or input/output space accessed. Address lines A31–A4 are used to drive addresses into the processor to perform cache line invalidations. Input signals must meet setup and hold times t_{22} and t_{23}. A31–A2 are not driven during bus or address hold.
BE0–3#	O	The **Byte Enable** signals indicate active bytes during read and write cycles. During the first cycle of a cache fill, the external system should assume that all byte enables are active. BE3# applies to D24–D31, BE2# applies to D16–D23, BE1# applies to D8–D15 and BE0# applies to D0–D7. BE0#–BE3# are active LOW and are not driven during bus hold.
DATA BUS		
D31–D0	I/O	The **Data Lines,** D0–D7, define the least significant byte of the data bus while lines D24–D31 define the most significant byte of the data bus. These signals must meet setup and hold times t_{22} and t_{23} for proper operation on reads. These pins are driven during the second and subsequent clocks of write cycles.
DATA PARITY		
DP0–DP3	I/O	There is one **Data Parity** pin for each byte of the data bus. Data parity is generated on all write data cycles with the same timing as the data driven by the Military Intel486 processor. Even parity information must be driven back into the processor on the data parity pins with the same timing as read information to insure that the correct parity check status is indicated by the Military Intel486 processor. The signals read on these pins do not affect program execution. Input signals must meet setup and hold times t_{22} and t_{23}. DP0–DP3 should be connected to V_{CC} through a pull-up resistor in systems that do not use parity. DP0–DP3 are active HIGH and are driven during the second and subsequent clocks of write cycles.
PCHK#	O	**Parity Status** is driven on the PCHK# pin the clock after ready for read operations. The parity status is for data sampled at the end of the previous clock. A parity error is indicated by PCHK# being LOW. Parity status is only checked for enabled bytes as indicated by the byte enable and bus size signals. PCHK# is valid only in the clock immediately after read data is returned to the processor. At all other times PCHK# is inactive (HIGH). PCHK# is never floated.

Table 3-3. Military Intel486™ Processor Pin Descriptions (Continued)

Symbol	Type	Name and Function
BUS CYCLE DEFINITION		
M/IO# **D/C#** **W/R#**	O O O	The **memory/input-output, data/control** and **write/read** lines are the primary bus definition signals. These signals are driven valid as the ADS# signal is asserted. **M/IO# D/C# W/R# — Bus Cycle Initiated** 0 0 0 Interrupt Acknowledge 0 0 1 Halt/Special Cycle 0 1 0 I/O Read 0 1 1 I/O Write 1 0 0 Code Read 1 0 1 Reserved 1 1 0 Memory Read 1 1 1 Memory Write The bus definition signals are not driven during bus hold and follow the timing of the address bus. Refer to section 10.2.11, ''Special Bus Cycles,'' for a description of the special bus cycles.
LOCK#	O	The **Bus Lock** pin indicates that the current bus cycle is locked. The Military Intel486 processor will not allow a bus hold when LOCK# is asserted (but address holds are allowed). LOCK# goes active in the first clock of the first locked bus cycle and goes inactive after the last clock of the last locked bus cycle. The last locked cycle ends when ready is returned. LOCK# is active LOW and is not driven during bus hold. Locked read cycles will not be transformed into cache fill cycles if KEN# is returned active.
PLOCK#	O	The **Pseudo-Lock** pin indicates that the current bus transaction requires more than one bus cycle to complete. For the Military Intel486 processor, examples of such operations are segment table descriptor reads (64 bits), in addition to cache line fills (128 bits). The Military Intel486 processor will drive PLOCK# active until the addresses for the last bus cycle of the transaction have been driven regardless of whether RDY# or BRDY# have been returned. Normally PLOCK# and BLAST# are inverse of each other. However during the first bus cycle of a 64-bit floating point write (for Military Intel486 processors with on-chip FPU), both PLOCK# and BLAST# will be asserted. PLOCK# is a function of the BS8#, BS16# and KEN# inputs. PLOCK# should be sampled only in the clock RDY# is returned. PLOCK# is active LOW and is not driven during bus hold.

Table 3-3. Military Intel486™ Processor Pin Descriptions (Continued)

Symbol	Type	Name and Function
BUS CONTROL		
ADS#	O	The **Address Status** output indicates that a valid bus cycle definition and address are available on the cycle definition lines and address bus. ADS# is driven active in the same clock as the addresses are driven. ADS# is active LOW and is not driven during bus hold.
RDY#	I	The **Non-burst Ready** input indicates that the current bus cycle is complete. RDY# indicates that the external system has presented valid data on the data pins in response to a read or that the external system has accepted data from the Military Intel486 processor in response to a write. RDY# is ignored when the bus is idle and at the end of the first clock of the bus cycle. RDY# is active during address hold. Data can be returned to the processor while AHOLD is active. RDY# is active LOW, and is not provided with an internal pull-up resistor. RDY# must satisfy setup and hold times t_{16} and t_{17} for proper chip operation.
BURST CONTROL		
BRDY#	I	The **Burst Ready** input performs the same function during a burst cycle that RDY# performs during a non-burst cycle. BRDY# indicates that the external system has presented valid data in response to a read or that the external system has accepted data in response to a write. BRDY# is ignored when the bus is idle and at the end of the first clock in a bus cycle. BRDY# is sampled in the second and subsequent clocks of a burst cycle. The data presented on the data bus will be strobed into the processor when BRDY# is sampled active. If RDY# is returned simultaneously with BRDY#, BRDY# is ignored and the burst cycle is prematurely aborted. BRDY# is active LOW and is provided with a small pull-up resistor. BRDY# must satisfy the setup and hold times t_{16} and t_{17}.
BLAST#	O	The **Burst Last** signal indicates that the next time BRDY# is returned the burst bus cycle is complete. BLAST# is active for both burst and non-burst bus cycles. BLAST# is active LOW and is not driven during bus hold.
INTERRUPTS		
RESET	I	The **Reset** input forces the Military Intel486 processor to begin execution at a known state. The processor cannot begin execution of instructions until at least 1 ms after V_{CC} and CLK have reached their proper DC and AC specifications. The RESET pin should remain active during this time to insure proper processor operation. RESET is active HIGH. RESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.

Table 3-3. Military Intel486™ Processor Pin Descriptions (Continued)

Symbol	Type	Name and Function
INTERRUPTS (Continued)		
INTR	I	The **Maskable Interrupt** indicates that an external interrupt has been generated. If the internal interrupt flag is set in EFLAGS, active interrupt processing will be initiated. The Military Intel486 processor will generate two locked interrupt acknowledge bus cycles in response to the INTR pin going active. INTR must remain active until the interrupt acknowledges have been performed to assure that the interrupt is recognized. INTR is active HIGH and is not provided with an internal pull-down resistor. INTR is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
NMI	I	The **Non-Maskable Interrupt** request signal indicates that an external non-maskable interrupt has been generated. NMI is rising edge sensitive. NMI must be held LOW for at least four CLK periods before this rising edge. NMI is not provided with an internal pull-down resistor. NMI is asynchronous, but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
SRESET	I	The **Soft Reset pin** duplicates all the functionality of the RESET pin with the following exception: 1. The SMBASE register will retain its previous value. For soft resets, SRESET should remain active for at least 15 CLK periods. SRESET is active HIGH. SRESET is asynchronous but must meet setup and hold times t_{20} and t_{21} for recognition in any specific clock.
SMI#	I	The **System Management Interrupt** input is used to invoke the System Management Mode (SMM). SMI# is a falling edge triggered signal which forces the processor into SMM at the completion of the current instruction. SMI# is recognized on an instruction boundary and at each iteration for repeat string instructions. SMI# does not break LOCKed bus cycles and cannot interrupt a currently executing SMM. The processor will latch the falling edge of one pending SMI# signal while the processor is executing an existing SMI#. The nested SMI# will not be recognized until after the execution of a Resume (RSM) instruction.
SMIACT#	O	The **System Management Interrupt ACTive** is an active low output, indicating that the processor is operating in SMM. It is asserted when the processor begins to execute the SMI# state save sequence and will remain active LOW until the processor executes the last state restore cycle out of SMRAM.
STPCLK#	I	The **SToP CLocK request** input signal indicates a request has been made to turn off the CLK input. When the processor recognizes a STPCLK#, the processor will stop execution on the next instruction boundary, unless superseded by a higher priority interrupt, empty all internal pipelines and the write buffers and generate a Stop Grant acknowledge bus cycle. STPCLK# is active LOW and is provided with an internal pull-up resistor. **STPCLK# is an asynchronous signal, but must remain active until the processor issues the Stop Grant bus cycle. STPCLK# may be de-asserted at any time after the processor has issued the Stop Grant bus cycle.**

Table 3-3. Military Intel486™ Processor Pin Descriptions (Continued)

Symbol	Type	Name and Function
BUS ARBITRATION		
BREQ	O	The **Bus Request** signal indicates that the Military Intel486 processor has internally generated a bus request. BREQ is generated whether or not the Military Intel486 processor is driving the bus. BREQ is active HIGH and is never floated.
HOLD	I	The **Bus Hold** *request* allows another bus master complete control of the processor bus. In response to HOLD going active the Military Intel486 processor will float most of its output and input/output pins. HLDA will be asserted after completing the current bus cycle, burst cycle or sequence of locked cycles. The Military Intel486 processor will remain in this state until HOLD is de-asserted. HOLD is active high and is not provided with an internal pull-down resistor. HOLD must satisfy setup and hold times t_{18} and t_{19} for proper operation.
HLDA	O	**Hold Acknowledge** goes active in response to a hold request presented on the HOLD pin. HLDA indicates that the Military Intel486 processor has given the bus to another local bus master. HLDA is driven active in the same clock that the Military Intel486 processor floats its bus. HLDA is driven inactive when leaving bus hold. HLDA is active HIGH and remains driven during bus hold.
BOFF#	I	The **Backoff** input forces the Military Intel486 processor to float its bus in the next clock. The processor will float all pins normally floated during bus hold but HLDA will not be asserted in response to BOFF#. BOFF# has higher priority than RDY# or BRDY#; if both are returned in the same clock, BOFF# takes effect. The processor remains in bus hold until BOFF# is negated. If a bus cycle was in progress when BOFF# was asserted the cycle will be restarted. BOFF# is active LOW and must meet setup and hold times t_{18} and t_{19} for proper operation.
CACHE INVALIDATION		
AHOLD	I	The **Address Hold** request allows another bus master access to the processor's address bus for a cache invalidation cycle. The Military Intel486 processor will stop driving its address bus in the clock following AHOLD going active. Only the address bus will be floated during address hold, the remainder of the bus will remain active. AHOLD is active HIGH and is provided with a small internal pull-down resistor. For proper operation AHOLD must meet setup and hold times t_{18} and t_{19}.
EADS#	I	This signal indicates that a *valid* **External Address** has been driven onto the Military Intel486 processor address pins. This address will be used to perform an internal cache invalidation cycle. EADS# is active LOW and is provided with an internal pull-up resistor. EADS# must satisfy setup and hold times t_{12} and t_{13} for proper operation.
CACHE CONTROL		
KEN#	I	The **Cache Enable** pin is used to determine whether the current cycle is cacheable. When the Military Intel486 processor generates a cycle that can be cached and KEN# is active one clock before RDY# or BRDY# during the first transfer of the cycle, the cycle will become a cache line fill cycle. Returning KEN# active one clock before RDY# during the last read in the cache line fill will cause the line to be placed in the on-chip cache. KEN# is active LOW and is provided with a small internal pull-up resistor. KEN# must satisfy setup and hold times t_{14} and t_{15} for proper operation.
FLUSH#	I	The **Cache Flush** input forces the Military Intel486 processor to flush its entire internal cache. FLUSH# is active low and need only be asserted for one clock. FLUSH# is asynchronous but setup and hold times t_{20} and t_{21} must be met for recognition in any specific clock.

Table 3-3. Military Intel486™ Processor Pin Descriptions (Continued)

Symbol	Type	Name and Function
PAGE CACHEABILITY		
PWT **PCD**	O O	The **Page Write-Through** and **Page Cache Disable** pins reflect the state of the page attribute bits, PWT and PCD, in the page table entry, page directory entry or control register 3 (CR3) when paging is enabled. If paging is disabled, the processor ignores the PCD and PWT bits and assumes they are zero for the purpose of caching and driving PCD and PWT pins. PWT and PCD have the same timing as the cycle definition pins (M/IO#, D/C#, and W/R#). PWT and PCD are active HIGH and are not driven during bus hold. PCD is masked by the cache disable bit (CD) in Control Register 0.
BUS SIZE CONTROL		
BS16# **BS8#**	I I	The **Bus Size 16** and **Bus Size 8** pins (bus sizing pins) cause the Military Intel486 processor to run multiple bus cycles to complete a request from devices that cannot provide or accept 32 bits of data in a single cycle. The bus sizing pins are sampled every clock. The state of these pins in the clock before ready is used by the Military Intel486 processor to determine the bus size. These signals are active LOW and are provided with internal pull-up resistors. These inputs must satisfy setup and hold times t_{14} and t_{15} for proper operation.
ADDRESS MASK		
A20M#	I	When the **Address Bit 20 Mask** pin is asserted, the Military Intel486 processor masks physical address bit 20 (A20) before performing a lookup to the internal cache or driving a memory cycle on the bus. A20M# emulates the address wraparound at one Mbyte, which occurs on the 8086 processor. A20M# is active LOW and should be asserted only when the processor is in real mode. This pin is asynchronous but should meet setup and hold times t_{20} and t_{21} for recognition in any specific clock. For proper operation, A20M# should be sampled high at the falling edge of RESET.
TEST ACCESS PORT		
TCK	I	**Test ClocK** is an input to the Military Intel486 processor and provides the clocking function required by the JTAG Boundary scan feature. TCK is used to clock state information and data into component on the rising edge of TCK on TMS and TDI, respectively. Data is clocked out of the part on the falling edge of TCK and TDO. TCK is provided with an internal pull-up resistor.
TDI	I	**Test Data Input** is the serial input used to shift JTAG instructions and data into component. TDI is sampled on the rising edge of TCK, during the SHIFT-IR and SHIFT-DR TAP controller states. During all other tap controller states, TDI is a ‘‘don’t care.’’ TDI is provided with an internal pull-up resistor.
TDO	O	**Test Data Output** is the serial output used to shift JTAG instructions and data out of the component. TDO is driven on the falling edge of TCK during the SHIFT-IR and SHIFT-DR TAP controller states. At all other times TDO is driven to the high impedance state.
TMS	I	**Test Mode Select** is decoded by the JTAG TAP (Tap Access Port) to select the operation of the test logic. TMS is sampled on the rising edge of TCK. To guarantee deterministic behavior of the TAP controller TMS is provided with an internal pull-up resistor.

Table 3-3. Military Intel486™ Processor Pin Descriptions (Continued)

Symbol	Type	Name and Function
NUMERIC ERROR REPORTING FOR MILITARY INTEL486 DX, INTELDX2™, AND INTELDX4™ PROCESSORS		
FERR#	O	The **Floating point ERRor** pin is driven active when a floating point error occurs. FERR# is similar to the ERROR# pin on the Intel387™ Math CoProcessor. FERR# is included for compatibility with systems using DOS type floating point error reporting. FERR# will not go active if FP errors are masked in FPU register. FERR# is active LOW, and is not floated during bus hold.
IGNNE#	I	When the **IGNore Numeric Error** pin is asserted the processor will ignore a numeric error and continue executing non-control floating point instructions, but FERR# will still be activated by the processor. When IGNNE# is de-asserted the processor will freeze on a non-control floating point instruction, if a previous floating point instruction caused an error. IGNNE# has no effect when the NE bit in control register 0 is set. IGNNE# is active LOW and is provided with a small internal pull-up resistor. IGNNE# is asynchronous but setup and hold times t_{20} and t_{21} must be met to insure recognition on any specific clock.
INTELDX4 PROCESSOR CLKMUL, VCC5, AND VOLDET		
CLKMUL	I	The **CLocK MULtiplier** input, defined during device RESET, defines the ratio of internal core frequency to external bus frequency. If sampled low, the core frequency operates at twice the external bus frequency (speed doubled mode). If driven high or left floating, speed triple mode is selected. CLKMUL has an internal pull-up speed to V_{CC} and may be left floating in designs that select speed tripled clock mode.
V_{CC5}	I	The **5V reference voltage** input is the reference voltage for the 5V-tolerant I/O buffers. This signal should be connected to $+5V \pm 5\%$ for use with 5V logic. If all inputs are from 3V logic, this pin should be connected to 3.3V.
VOLDET	O	A **VOLtage DETect** signal allows external system logic to distinguish between a 5V Military Intel486 processor and the 3.3V IntelDX4 processor. This signal is active low for a 3.3V IntelDX4 processor.

Table 3-4. Output Pins

Name	Active Level	When Floated
BREQ	HIGH	
HLDA	HIGH	
BE3#–BE0#	LOW	Bus Hold
PWT, PCD	HIGH/LOW	Bus Hold
W/R#, M/IO#, D/C#	HIGH/LOW	Bus Hold
LOCK#	LOW	Bus Hold
PLOCK#	LOW	Bus Hold
ADS#	LOW	Bus Hold
BLAST#	LOW	Bus Hold
PCHK#	LOW	
FERR#	LOW	
A3–A2	N/A	Bus, Address Hold
SMIACT#	LOW	
VOLDET(1)	LOW	

NOTE:
1. Present on the IntelDX4 processor only.

Table 3-5. Input/Output Pins

Name	Active Level	When Floated
D31–D0	HIGH/LOW	Bus Hold
DP3–DP0	HIGH	Bus Hold
A31–A4	HIGH/LOW	Bus, Address Hold

Table 3-6. Test Pins

Name	Input or Output	Sampled/Driven On
TCK	Input	N/A
TDI	Input	Rising Edge of TCK
TDO	Output	Falling Edge of TCK
TMS	Input	Rising Edge of TCK

Table 3-7. Input Pins

Name	Active Level	Synchronous/ Asynchronous	Internal Pull-Up/ Pull-Down
CLK			
RESET	HIGH	Asynchronous	
SRESET	HIGH	Asynchronous	Pull-Down
HOLD	HIGH	Synchronous	
AHOLD	HIGH	Synchronous	Pull-Down
EADS#	LOW	Synchronous	Pull-Up
BOFF#	LOW	Synchronous	Pull-Up
FLUSH#	LOW	Asynchronous	Pull-Up
A20M#	LOW	Asynchronous	Pull-Up
BS16#, BS8#	LOW	Synchronous	Pull-Up
KEN#	LOW	Synchronous	Pull-Up
RDY#	LOW	Synchronous	
BRDY#	LOW	Synchronous	Pull-Up
INTR	HIGH	Asynchronous	
NMI	HIGH	Asynchronous	
IGNNE#	LOW	Asynchronous	Pull-Up
SMI#	LOW	Asynchronous	Pull-Up
STPCLK#	LOW	Asynchronous	Pull-Up
TCK	HIGH		Pull-Up
TDI	HIGH		Pull-Up
TMS	HIGH		Pull-Up
CLKMUL# (1)	N/A		Pull-Up

NOTE:
1. Present on the IntelDX4 processor only.

4.0 ARCHITECTURAL OVERVIEW

4.1 Introduction

The Military Intel486 processor family is a 32-bit architecture with on-chip memory management, floating point, and cache memory units. Figure 4-1 is a block diagram of the Military Intel486 processor family. The Military Intel486 processor contains all the features of the Intel386™ processor with enhancements to increase performance.

The Military Intel486 processor instruction set includes the complete Intel386 processor instruction set along with extensions to serve new applications and increase performance. The on-chip memory management unit (MMU) is completely compatible with the Intel386 processor MMU. Software written for previous members of the Intel architecture family will run on the Military Intel486 processor without any modifications.

On-chip cache memory allows frequently used data and code to be stored on-chip reducing accesses to the external bus. RISC design techniques reduce instruction cycle times. A burst bus feature enables fast cache fills.

The memory management unit (MMU) consists of a segmentation unit and a paging unit. Segmentation allows management of the logical address space by providing easy data and code relocatibility and efficient sharing of global resources. The paging mechanism operates beneath segmentation and is transparent to the segmentation process. Paging is optional and can be disabled by system software. Each segment can be divided into one or more 4-Kbyte segments. To implement a virtual memory system, full restartability for all page and segment faults is supported.

Memory is organized into one or more variable length segments, each up to four Gbytes (232 bytes) in size. A segment can have attributes associated with it which include its location, size, type (i.e., stack, code or data), and protection characteristics. Each task on a Military Intel486 processor can have a maximum of 16,381 segments and each are up to four Gbytes in size. Thus, each task has a maximum of 64 terabytes (trillion bytes) of virtual memory.

The segmentation unit provides four levels of protection for isolating and protecting applications and the operating system from each other. The hardware enforced protection allows the design of systems with a high degree of software integrity.

The Military Intel486 processor has two modes of operation: Real Address Mode (Real Mode) and Protected Mode Virtual Address Mode (Protected Mode). In Real Mode the Military Intel486 processor operates as a very fast 8086. Real Mode is required primarily to set up the Military Intel486 processor for Protected Mode operation. Protected Mode provides access to the sophisticated memory management paging and privilege capabilities of the processor.

Within Protected Mode, software can perform a task switch to enter into tasks designated as Virtual 8086 Mode tasks. Each Virtual 8086 task behaves with 8086 semantics, allowing 8086 processor software (an application program or an entire operating system) to execute.

System Management Mode (SMM) provides the system designer with a means of adding new software controlled features to their computer products that always operate transparently to the Operating System (OS) and software applications. SMM is intended for use only by system firmware, not by applications software or general purpose systems software.

The on-chip cache is 16 Kbytes in size for the IntelDX4 processor and 8 Kbytes in size for all other members of the Military Intel486 processor family. It is 4-way set associative and follows a write-through policy. The on-chip cache includes features to provide flexibility in external memory system design. Individual pages can be designated as cacheable or non-cacheable by software or hardware. The cache can also be enabled and disabled by software or hardware.

The Military Intel486 processor also has features that facilitate high-performance hardware designs. The 1X bus clock input eases high-frequency board-level designs. The clock multiplier on IntelDX2 and IntelDX4 processors improves execution performance without increasing board design complexity. The clock multiplier enhances all operations operating out of the cache and/or not blocked by external bus accesses. The burst bus feature enables fast cache fills.

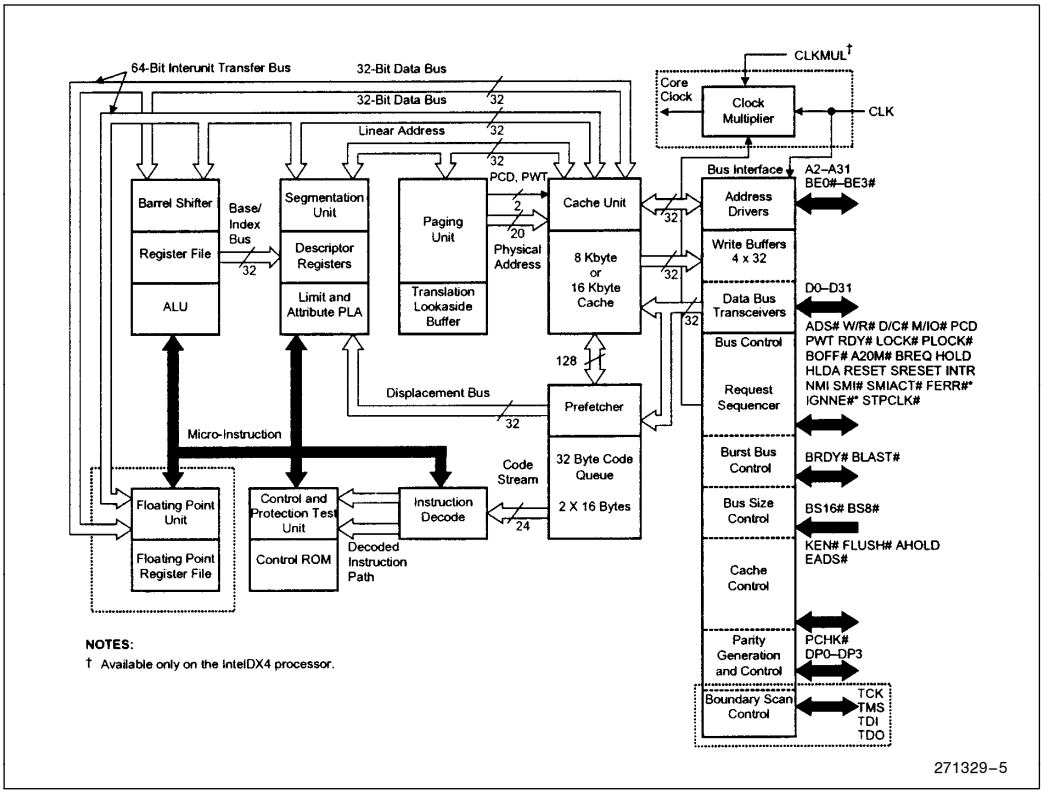


Figure 4-1. Military Intel486™ Processor Block Diagram

4.1.1 MILITARY INTEL486 DX, INTELDX2™, AND INTELDX4™ PROCESSOR ON-CHIP FLOATING POINT UNIT

The Military Intel486 DX, IntelDX2, and IntelDX4 processors incorporate the basic Military Intel486 processor 32-bit architecture with on-chip memory management and cache memory units. They also have an on-chip floating point unit (FPU) that operates in parallel with the arithmetic and logic unit. The FPU provides arithmetic instructions for a variety of numeric data types and executes numerous built-in transcendental functions (e.g., tangent, sine, cosine, and log functions). The floating point unit fully conforms to the ANSI/IEEE standard 754-1985 for floating point arithmetic.

All software written for the Intel386 processor, Intel387 math coprocessor and previous members of the 86/87 architectural family will run on these processors without any modifications.

4.2 Register Set

The Military Intel486 processor register set can be split into the following categories:

- Base Architecture Registers
  - — General Purpose Registers
  - — Instruction Pointer
  - — Flags Register
  - — Segment Registers

- Systems Level Registers
  - — Control Registers
  - — System Address Registers
- Debug and Test Registers

The base architecture and floating point registers (see below) are accessible by the applications program. The system level registers can only be accessed at privilege level 0 and used by system level programs. The debug and test registers also can only be accessed at privilege level 0.

4.2.1 FLOATING POINT REGISTERS

In addition to the registers listed above, the Military Intel486 DX, IntelDX2, and IntelDX4 processors also have the following:

- Floating Point Registers
  - — Data Registers
  - — Tag Word
  - — Status Word
  - — Instruction and Data Pointers
  - — Control Word

4.2.2 BASE ARCHITECTURE REGISTERS

Figure 4-2 shows the Military Intel486 processor base architecture registers. The contents of these registers are task-specific and are automatically loaded with a new context upon a task switch operation.

The base architecture includes six directly accessible descriptors, each specifying a segment up to 4 Gbytes in size. The descriptors are indicated by the selector values placed in the Military Intel486 processor segment registers. Various selector values can be loaded as a program executes.

The selectors are also task-specific, so the segment registers are automatically loaded with new context upon a task switch operation.

NOTE:
In register descriptions, ''set'' means ''set to 1,'' and ''reset'' means ''reset to 0.''

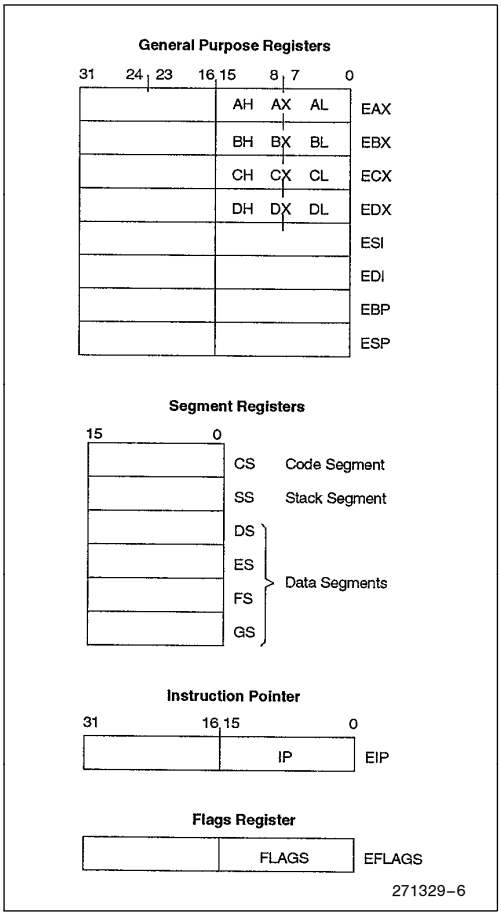


Figure 4-2. Base Architecture Registers

4.2.2.1 General Purpose Registers

The eight 32-bit general purpose registers are shown in Figure 4-2. These registers hold data or address quantities. The general purpose registers can support data operands of 1, 8, 16 and 32 bits, and bit fields of 1 to 32 bits. Address operands of 16 and 32 bits are supported. The 32-bit registers are named EAX, EBX, ECX, EDX, ESI, EDI, EBP and ESP.

The least significant 16 bits of the general purpose registers can be accessed separately by using the 16-bit names of the registers AX, BX, CX, DX, SI, DI, BP and SP. The upper 16 bits of the register are not changed when the lower 16 bits are accessed separately.

Finally, 8-bit operations can individually access the lower byte (bits 0–7) and the highest byte (bits 8–15) of the general purpose registers AX, BX, CX and DX. The lowest bytes are named AL, BL, CL and DL respectively. The higher bytes are named AH, BH, CH and DH respectively. The individual byte accessibility offers additional flexibility for data operations, but is not used for effective address calculation.

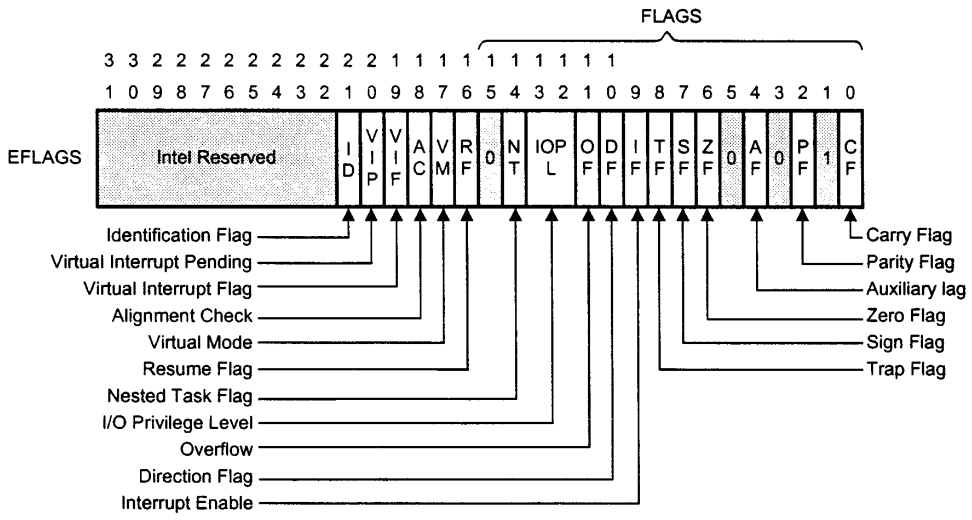
4.2.2.2 Instruction Pointer

The instruction pointer shown in Figure 4-2 is a 32-bit register named EIP. EIP holds the offset of the next instruction to be executed. The offset is always relative to the base of the code segment (CS). The lower 16 bits (bits 0–15) of the EIP contain the 16-bit instruction pointer named IP, which is used for 16-bit addressing.

4.2.2.3 Flags Register

The flags register is a 32-bit register named EFLAGS. The defined bits and bit fields within EFLAGS control certain operations and indicate status of the Military Intel486 processor. The lower 16 bits (bit 0–15) of EFLAGS contain the 16-bit register named FLAGS, which is most useful when executing 8086 and 80286 processor code. EFLAGS is shown in Figure 4-3.

EFLAGS bits 1, 3, 5, 15 and 22–31 are defined as "Intel Reserved." When these bits are stored during interrupt processing or with a PUSHF instruction (push flags onto stack), a one is stored in bit 1 and zeros in bits 3, 5, 15 and 22–31.



NOTE:
See section 4.2.7 "Compatibility."

Figure 4-3. Flag Registers

ID (Identification Flag, bit 21)

The ability of a program to set and clear the ID flag indicates that the processor supports the CPUID instruction. (Refer to section 13, ''Instruction Set Summary,'' and Appendix A, ''Feature Determination: CPUID Instruction.'')

VIP (Virtual Interrupt Pending Flag, bit 20)

The VIP flag together with the VIF enable each applications program in a multitasking environment to have virtualized versions of the system's IF flag.

VIF (Virtual Interrupt Flag, bit 19)

The VIF is a virtual image of IF (the interrupt flag) used with VIP.

AC (Alignment Check, bit 18)

The AC bit is defined in the upper 16 bits of the register. It enables the generation of faults if a memory reference is to a misaligned address. Alignment faults are enabled when AC is set to 1. A misaligned address is a word access an odd address, a dword access to an address that is not on a dword boundary, or an 8-byte reference to an address that is not on a 64-bit word boundary. (See section 10.1.5, ''Operand Alignment.'')

Alignment faults are only generated by programs running at privilege level 3. The AC bit setting is ignored at privilege levels 0, 1 and 2. Note that references to the descriptor tables (for selector loads), or the task state segment (TSS), are implicitly level 0 references even if the instructions causing the references are executed at level 3. Alignment faults are reported through interrupt 17, with an error code of 0. Table 4-1 gives the alignment required for the Military Intel486 processor data types.

Table 4-1. Data Type Alignment Requirements

Memory Access	Alignment (Byte Boundary)
Word	2
Dword	4
Single Precision Real	4
Double Precision Real	8
Extended Precision Real	8
Selector	2
48-Bit Segmented Pointer	4
32-Bit Flat Pointer	4
32-Bit Segmented Pointer	2
48-Bit ''Pseudo-Descriptor''	4
FSTENV/FLDENV Save Area	4/2 (On Operand Size)
FSAVE/FRSTOR Save Area	4/2 (On Operand Size)
Bit String	4

IMPLEMENTATION NOTE:

Several instructions on the Military Intel486 processor generate misaligned references, even if their memory address is aligned. For example, on the Military Intel486 processor, the SGDT/SIDT (store global/interrupt descriptor table) instruction reads/writes two bytes, and then reads/writes four bytes from a ''pseudo-descriptor'' at the given address. The Military Intel486 processor will generate misaligned references unless the address is on a 2 mod 4 boundary. The FSAVE and FRSTOR instructions (floating point save and restore state) will generate misaligned references for one-half of the register save/restore cycles. The Military Intel486 processor will not cause any AC faults if the effective address given in the instruction has the proper alignment.

VM (Virtual 8086 Mode, bit 17)

The VM bit provides Virtual 8086 Mode within Protected Mode. If set while the Military Intel486 processor is in Protected Mode, the Military Intel486 processor will switch to Virtual 8086 operation, handling segment loads as the 8086 processor does, but generating exception 13 faults on privileged opcodes. The VM bit can be set only in Protected Mode, by the IRET instruction (if current privilege level = 0) and by task switches at any privilege level. The VM bit is unaffected by POPF. PUSHF always pushes a 0 in this bit, even if executing in Virtual 8086 Mode. The EFLAGS image pushed during interrupt processing or saved during task switches will contain a 1 in this bit if the interrupted code was executing as a Virtual 8086 Task.

RF (Resume Flag, bit 16)

The RF flag is used in conjunction with the debug register breakpoints. It is checked at instruction boundaries before breakpoint processing. When RF is set, it causes any debug fault to be ignored on the next instruction. RF is then automatically reset at the successful completion of every instruction (no faults are signaled) except the IRET instruction, the POPF instruction, (and JMP, CALL, and INT instructions causing a task switch). These instructions set RF to the value specified by the memory image. For example, at the end of the breakpoint service routine, the IRET instruction can pop an EFLAG image having the RF bit set and resume the program's execution at the breakpoint address without generating another breakpoint fault on the same location.

NT (Nested Task, bit 14)

The flag applies to Protected Mode. NT is set to indicate that the execution of this task is within another task. If set, it indicates that the current nested task's Task State Segment (TSS) has a valid back link to the previous task's TSS. This bit is set or reset by control transfers to other tasks. The value of NT in EFLAGS is tested by the IRET instruction to determine whether to do an inter-task return or an intra-task return. A POPF or an IRET instruction will affect the setting of this bit according to the image popped, at any privilege level.

IOPL (Input/Output Privilege Level, bits 12–13)

This two-bit field applies to Protected Mode. IOPL indicates the numerically maximum CPL (current privilege level) value permitted to execute I/O instructions without generating an exception 13 fault or consulting the I/O Permission Bitmap. It also indicates the maximum CPL value allowing alteration of the IF (INTR Enable Flag) bit when new values are popped into the EFLAG register. POPF and IRET instruction can alter the IOPL field when executed at CPL = 0. Task switches can always alter the IOPL field, when the new flag image is loaded from the incoming task's TSS.

OF (Overflow Flag, bit 11)

is set if the operation resulted in a signed overflow. Signed overflow occurs when the operation resulted in carry/borrow **into** the sign bit (high-order bit) of the result but did not result in a carry/borrow **out of** the high-order bit, or vice-versa. For 8-, 16-, 32-bit operations, OF is set according to overflow at bit 7, 15, 31, respectively.

DF (Direction Flag, bit 10)

DF defines whether ESI and/or EDI registers post decrement or post increment during the string instructions. Post increment occurs if DF is reset. Post decrement occurs if DF is set.

IF (INTR Enable Flag, bit 9)

IF flag, when set, allows recognition of external interrupts signaled on the INTR pin. When IF is reset, external interrupts signaled on the INTR are not recognized. IOPL indicates the maximum CPL value allowing alteration of the IF bit when new values are popped into EFLAGS or FLAGS.

TF (Trap Enable Flag, bit 8)

TF controls the generation of exception 1 trap when single-stepping through code. When TF is set, the Military Intel486 processor generates an exception 1 trap after the next instruction is executed. When TF is reset, exception 1 traps occur only as a function of the breakpoint addresses loaded into debug registers DR0–DR3.

SF (Sign Flag, bit 7)

SF is set if the high-order bit of the result is set, it is reset otherwise. For 8-, 16-, 32-bit operations, SF reflects the state of bit 7, 15, 31 respectively.

ZF (Zero Flag, bit 6)

ZF is set if all bits of the result are 0. Otherwise, it is reset.

AF (Auxiliary Carry Flag, bit 4)

The Auxiliary Flag is used to simplify the addition and subtraction of packed BCD quantities. AF is set if the operation resulted in a carry out of bit 3 (addition) or a borrow into bit 3 (subtraction). Otherwise, AF is reset. AF is affected by carry out of, or borrow into bit 3 only, regardless of overall operand length: 8, 16 or 32 bits.

PF (Parity Flags, bit 2)

PF is set if the low-order eight bits of the operation contains an even number of "1's" (even parity). PF is reset if the low-order eight bits have odd parity. PF is a function of only the low-order eight bits, regardless of operand size.

CF (Carry Flag, bit 0)

CF is set if the operation resulted in a carry out of (addition), or a borrow into (subtraction) the high-order bit. Otherwise, CF is reset. For 8-, 16- or 32-bit operations, CF is set according to carry/borrow at bit 7, 15 or 31, respectively.

4.2.2.4 Segment Registers

Six 16-bit segment registers hold segment selector values identifying the currently addressable memory segments. In protected mode, each segment may range in size from one byte up to the entire linear and physical address space of the machine, 4 Gbytes (2^{32} bytes). In real address mode, the maximum segment size is fixed at 64 Kbytes (2^{16} bytes).

The six addressable segments are defined by the segment registers CS, SS, DS, ES, FS and GS. The selector in CS indicates the current code segment; the selector in SS indicates the current stack segment; the selectors in DS, ES, FS and GS indicate the current data segments.

4.2.2.5 Segment Descriptor Cache Registers

The segment descriptor cache registers are not programmer visible, yet it is very useful to understand their content. A programmer invisible descriptor cache register is associated with each programmer-visible segment register, as shown by Figure 4-4. Each descriptor cache register holds a 32-bit base address, a 32-bit segment limit, and the other necessary segment attributes.

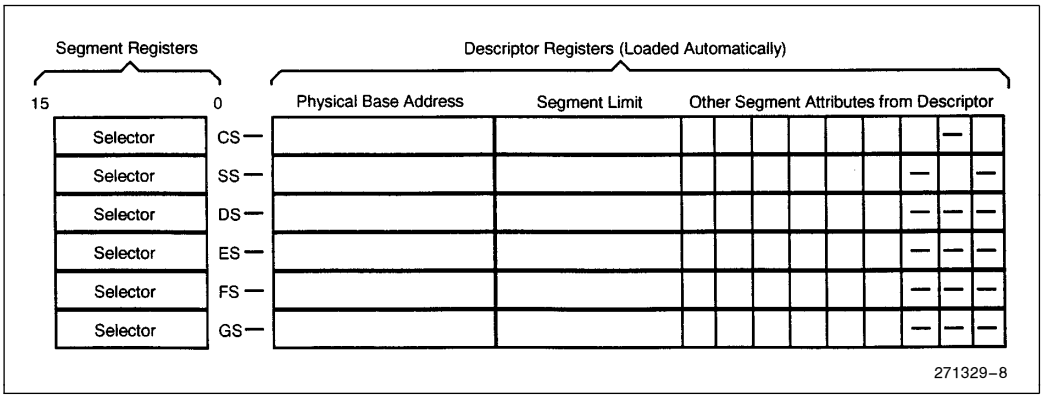


Figure 4-4. Military Intel486™ Processor Segment Registers and Associated Descriptor Cache Registers

When a selector value is loaded into a segment register, the associated descriptor cache register is automatically updated with the correct information. In Real Mode, only the base address is updated directly (by shifting the selector value four bits to the left), because the segment maximum limit and attributes are fixed in Real Mode. In Protected Mode, the base address, the limit, and the attributes are all updated per the contents of the segment descriptor indexed by the selector.

Whenever a memory reference occurs, the segment descriptor cache register associated with the segment being used is automatically involved with the memory reference. The 32-bit segment base address becomes a component of the linear address calculation, the 32-bit limit is used for the limit-check operation, and the attributes are checked against the type of memory reference requested.

4.2.3 SYSTEM LEVEL REGISTERS

Figure 4-5 illustrates the system level registers, which are the control operation of the on-chip cache, the on-chip floating point unit (on the Military Intel486 DX, IntelDX2, and IntelDX4 processors) and the segmentation and paging mechanisms. These registers are only accessible to programs running at privilege level 0, the highest privilege level.

The system level registers include three control registers and four segmentation base registers. The three control registers are CR0, CR2 and CR3. CR1 is reserved for future Intel processors. The four segmentation base registers are the Global Descriptor Table Register (GDTR), the Interrupt Descriptor Table Register (IDTR), the Local Descriptor Table Register (LDTR) and the Task State Segment Register (TR).

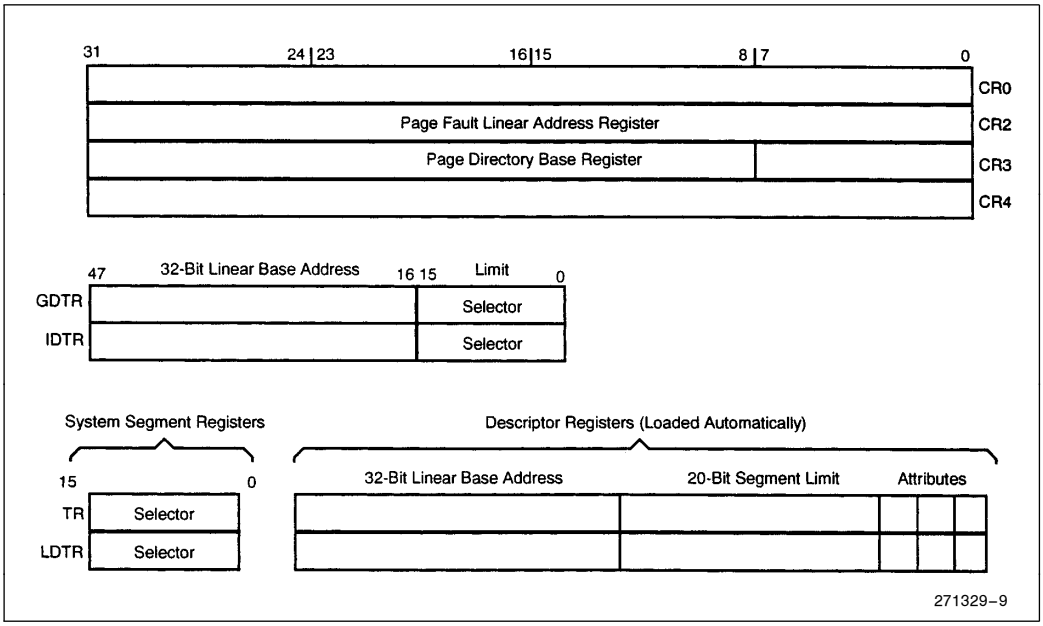
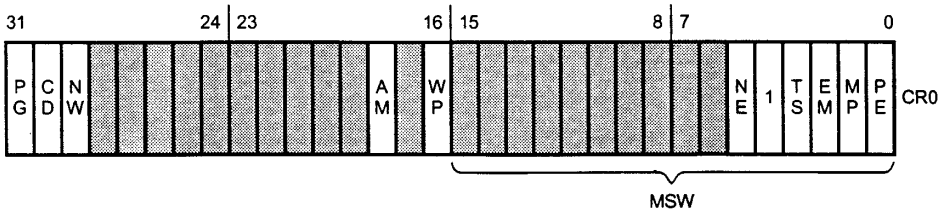


Figure 4-5. System Level Registers



NOTE:
See Section 4.2.7 ''Compatibility''.

Figure 4-6. Control Register 0

4.2.3.1 Control Registers

Control Register 0 (CR0)

CR0, shown in Figure 4-6, contains 10 bits for control and status purposes. The function of the bits in CR0 can be categorized as follows:

- Military Intel486 Processor Operating Modes: PG, PE (Table 4-2)
- On-Chip Cache Control Modes: CD, NW (Table 4-3)
- On-Chip Floating Point Unit: NE, TS, EM, TS (Tables 4-4 and 4-5).
- Alignment Check Control: AM
- Supervisor Write Protect: WP

Table 4-2. Military Intel486™ Processor Operating Modes

PG	PE	Mode
0	0	REAL Mode. Exact 8086 processor semantics, with 32-bit extensions available with prefixes.
0	1	Protected Mode. Exact 80286 processor semantics, plus 32-bit extensions through both prefixes and ''default'' prefix setting associated with code segment descriptors. Also, a sub-mode is defined to support a virtual 8086 processor within the context of the extended 80286 processor protection model.
1	0	UNDEFINED. Loading CR0 with this combination of PG and PE bits will raise a GP fault with error code 0.
1	1	Paged Protected Mode. All the facilities of Protected mode, with paging enabled underneath segmentation.

Table 4-3. On-Chip Cache Control Modes

CD	NW	Operating Mode
1	1	Cache fills disabled, write-through and invalidates disabled.
1	0	Cache fills disabled, write-through and invalidates enabled.
0	1	INVALID. If CR0 is loaded with this configuration of bits, a GP fault with error code is raised.
0	0	Cache fills enabled, write-through and invalidates enabled.

The low-order 16 bits of CR0 are also known as the Machine Status Word (MSW), for compatibility with the 80286 processor protected mode. LMSW and SMSW (load and store MSW) instructions are taken as special aliases of the load and store CR0 operations, where only the low-order 16 bits of CR0 are involved. The LMSW and SMSW instructions in the Military Intel486 processor work in an identical fashion to the LMSW and SMSW instructions in the 80286 processor (i.e., they only operate on the low-order 16 bits of CR0 and ignores the new bits). New Military Intel486 processor operating systems should use the MOV CR0, Reg instruction.

NOTE:

All Intel386 and Military Intel486 processor CR0 bits, except for ET and NE, are upwardly compatible with the 80286 processor, because they are in register bits not defined in the 80286 processor. For strict compatibility with the 80286 processor, the load machine status word (LMSW) instruction is defined to not change the ET or NE bits.

The defined CR0 bits are described below.

PG (Paging Enable, bit 31)

PG bit is used to indicate whether paging is enabled (PG = 1) or disabled (PG = 0). (See Table 4-2.)

CD (Cache Disable, bit 30)

The CD bit is used to enable the on-chip cache. When CD = 1, the cache will not be filled on cache misses. When CD = 0, cache fills may be performed on misses. (See Table 4-3.)

The state of the CD bit, the cache enable input pin (KEN#), and the relevant page cache disable (PCD) bit determine if a line read in response to a cache miss will be installed in the cache. A line is installed in the cache only if CD = 0 and KEN# and PCD are both zero. The relevant PCD bit comes from either the page table entry, page directory entry or control register 3. (Refer to section 7.6, ''Page Cacheability.'')

CD is set to one after RESET.

NW (Not Write-Through, bit 29)

The NW bit enables on-chip cache write-throughs and write-invalidate cycles (NW = 0).

When NW = 0, all writes, including cache hits, are sent out to the pins. Invalidate cycles are enabled when NW = 0. During an invalidate cycle a line will be removed from the cache if the invalidate address hits in the cache. (See Table 4-3.)

When NW = 1, write-throughs and write-invalidate cycles are disabled. A write will not be sent to the pins if the write hits in the cache. With NW = 1 the only write cycles that reach the external bus are cache misses. Write hits with NW = 1 will never update main memory. Invalidate cycles are ignored when NW = 1.

AM (Alignment Mask, bit 18)

The AM bit controls whether the alignment check (AC) bit in the flag register (EFLAGS) can allow an alignment fault. AM = 0 disables the AC bit. AM = 1 enables the AC bit. AM = 0 is the Intel386 processor compatible mode.

Intel386 processor software may load incorrect data into the AC bit in the EFLAGS register. Setting AM = 0 will prevent AC faults from occurring before the Military Intel486 processor has created the AC interrupt service routine.

WP (Write Protect, bit 16)

WP protects read-only pages from supervisor write access. The Intel386 processor allows a read-only page to be written from privilege levels 0–2. The Military Intel486 processor are compatible with the Intel386 processor when WP = 0. WP = 1 forces a fault on a write to a read-only page from any privilege level. Operating systems with Copy-on-Write features can be supported with the WP bit. (Refer to section 6.4.3 ''Page Level Protection (R/W, U/S Bits).'')

NOTE:

Refer to Tables 4-4 and 4-5 for values and interpolation of NE, EM, TS, and MP bits, in addition to the sections below.

NE (Numerics Exception, bit 5)

Military Intel486 DX, IntelDX2 and IntelDX4 Processor NE Bit

For Military Intel486 DX, IntelDX2, and IntelDX4 processors, the NE bit controls whether unmasked floating point exceptions (UFPE) are handled through interrupt vector 16 (NE = 1) or through an external interrupt (NE = 0). NE = 0 (default at reset) supports the DOS operating system error reporting scheme from the 8087, Intel287 and Intel387 math coprocessors. In DOS systems, math coprocessor errors are reported via external interrupt vector 13. DOS uses interrupt vector 16 for an operating system call. (Refer to sections 9.2.14, ''Numeric Error Reporting (FERR#, IGNNE#),'' and 10.2.14 ''Floating Point Error Handling.'')

For any UFPE, the floating point error output pin (FERR#) will be driven active.

For NE = 0, the Military Intel486 DX, IntelDX2 and IntelDX4 processors work in conjunction with the ignore numeric error input (IGNNE#) and the FERR# output pins. When a UFPE occurs and the IGNNE# input is inactive, the Military Intel486 DX, IntelDX2, and IntelDX4 processors freeze immediately before executing the next floating point instruction. An external interrupt controller will supply an interrupt vector when FERR# is driven active. The UFPE is ignored if IGNNE# is active and floating point execution continues.

NOTE:

The freeze does not take place if the next instruction is one of the control instructions FNCLEX, FNINIT, FNSAVE, FNSTENV, FNSTCW, FNSTSW, FNSTSW AX, FNENI, FNDISI and FNSETPM. The freeze does occur if the next instruction is WAIT.

For NE = 1, any UFPE will result in a software interrupt 16, immediately before executing the next non-control floating point or WAIT instruction. The ignore numeric error input (IGNNE#) signal will be ignored.

TS (Task Switch, bit 3)

Military Intel486 DX, IntelDX2, and IntelDX4 Processor TS Bit

For Military Intel486 DX, IntelDX2, and IntelDX4 processors, the TS bit is set whenever a task switch operation is performed. Execution of floating point instructions with TS = 1 will cause a Device Not Available (DNA) fault (trap vector 7). If TS = 1 and MP = 1 (monitor coprocessor in CR0), a WAIT instruction will cause a DNA fault.

EM (Emulate Coprocessor, bit 2)

Military Intel486 DX, IntelDX2, and IntelDX4 Processor EM Bit

For the Military Intel486 DX, IntelDX2, and IntelDX4 processors, the EM bit determines whether floating point instructions are trapped (EM = 1) or executed. If EM = 1, all floating point instructions will cause fault 7.

If EM = 0, the on-chip floating point will be used.

NOTE:

WAIT instructions are not affected by the state of EM. (See Table 4-5.)

MP (Monitor Coprocessor, bit 1)

Military Intel486 DX, IntelDX2, and IntelDX4 Processor MP Bit

For the Military Intel486 DX, IntelDX2, and IntelDX4 processors, the MP is used in conjunction with the TS bit to determine if WAIT instructions cause fault 7. (See Table 4-5.) The TS bit is set to 1 on task switches by the Military Intel486 DX, IntelDX2, and IntelDX4 processors. Floating point instructions are not affected by the state of the MP bit. It is recommended that the MP bit be set to one for normal processor operation.

PE (Protection Enable, bit 0)

The PE bit enables the segment based protection mechanism if PE = 1 protection is enabled. When PE = 0 the Military Intel486 processor operates in REAL mode, with segment based protection disabled, and addresses formed as in an 8086 processor. (Refer to Table 4-2.)

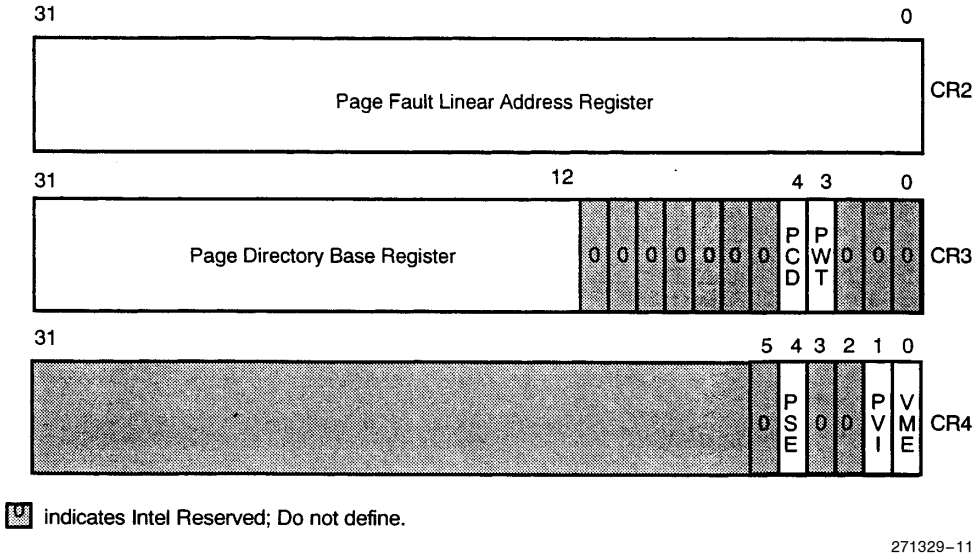
Table 4-4. Recommended Values of the Floating Point Related Bits for All Military Intel486™ Processors

CR0 Bit	Military Intel486 DX, IntelDX2™, and IntelDX4™ Processors
EM	0
MP	1
NE	0, for DOS Systems 1, for User-Defined Exception Handler

Table 4-5. Interpretation of Different Combinations of the EM, TS and MP Bits for All Military Intel486™ Processors

CR0 Bit			Instruction Type	
EM	TS	MP	Floating Point	Wait
0	0	0	Execute	Execute
0	0	1	Execute	Execute
0	1	0	Exception 7	Execute
0	1	1	Exception 7	Exception 7
1	0	0	Exception 7	Execute
1	0	1	Exception 7	Execute
1	1	0	Exception 7	Execute
1	1	1	Exception 7	Exception 7

NOTE:
For Military Intel486 DX, IntelDX2™ and IntelDX4™ processors, if MP = 1 and TS = 1, the processor will generate a trap 7 so that the system software can save the floating point status of the old task.



NOTE:
See section 4.2.7, "Compatibility."

Figure 4-7. Control Registers 2, 3 and 4

Control Register 1 (CR1)

CR1 is reserved for use in future Intel processors.

Control Register 2 (CR2)

CR2, shown in Figure 4-7, holds the 32-bit linear address that caused the last page fault detected. The error code pushed onto the page fault handler's stack when it is invoked provides additional status information on this page fault.

Control Register 3 (CR3)

CR3, shown in Figure 4-7, contains the physical base address of the page directory table. The page directory is always page aligned (4 Kbyte-aligned). This alignment is enforced by only storing bits 12–31 in CR3.

In the Military Intel486 processor, CR3 contains two bits, page write-through (PWT) (bit 3) and page cache disable (PCD) (bit 4). The page table entry (PTE) and page directory entry (PDE) also contain PWT and PCD bits. PWT and PCD control page cacheability. When a page is accessed in external memory, the state of PWT and PCD are driven out on the PWT and PCD pins. The source of PWT and PCD can be CR3, the PTE or the PDE. PWT and PCD are sourced from CR3 when the PDE is being updated. When paging is disabled (PG = 0 in CR0), PCD and PWT are assumed to be 0, regardless of their state in CR3.

A task switch through a task state segment (TSS) which changes the values in CR3, or an explicit load into CR3 with any value, will invalidate all cached page table entries in the translation lookaside buffer (TLB).

The page directory base address in CR3 is a physical address. The page directory can be paged out while its associated task is suspended, but the operating system must ensure that the page directory is resident in physical memory before the task is dispatched. The entry in the TSS for CR3 has a physical address, with no provision for a present bit. This means that the page directory for a task must be resident in physical memory. The CR3 image in a TSS must point to this area, before the task can be dispatched through its TSS.

Control Register 4 (CR4)

CR4, shown in Figure 4-7, contains bits that enable virtual mode extensions and protected mode virtual interrupts.

VME (Virtual-8086 Mode Extensions, bit 0 of CR4)

Setting this bit to 1 enables support for a virtual interrupt flag in virtual-8086 mode. This feature can improve the performance of virtual-8086 applications by eliminating the overhead of faulting to a virtual-8086 monitor for emulation of certain operations.

PVI (Protected-Mode Virtual Interrupts, bit 1 of CR4)

Setting this bit to 1 enables support for a virtual interrupt flag in protected mode. This feature can enable some programs designed for execution at privilege level 0 to execute at privilege level 3.

4.2.3.2 System Address Registers

Four special registers are defined to reference the tables or segments supported by the 80286, Intel386, and Military Intel486 processors' protection model. These tables or segments are: GDT (Global Descriptor Table), IDT (Interrupt Descriptor Table), LDT (Local Descriptor Table), TSS (Task State Segment).

The addresses of these tables and segments are stored in special registers, the System Address and System Segment Registers, illustrated in Figure 4-5. These registers are named GDTR, IDTR, LDTR and TR respectively. Section 6, ''Protected Mode Architecture,'' describes how to use these registers.

System Address Registers: GDTR and IDTR

The GDTR and IDTR hold the 32-bit linear-base address and 16-bit limit of the GDT and IDT, respectively.

Because the GDT and IDT segments are global to all tasks in the system, the GDT and IDT are defined by 32-bit linear addresses (subject to page translation if paging is enabled) and 16-bit limit values.

System Segment Registers: LDTR and TR

The LDTR and TR hold the 16-bit selector for the LDT descriptor and the TSS descriptor, respectively.

Because the LDT and TSS segments are task specific segments, the LDT and TSS are defined by selector values stored in the system segment registers.

NOTE:
A programmer-invisible segment descriptor register is associated with each system segment register.

4.2.4 FLOATING POINT REGISTERS

Figure 4-8 shows the floating point register set. The on-chip FPU contains eight data registers, a tag word, a control register, a status register, an instruction pointer and a data pointer.

The operation of the Military Intel486 DX, IntelDX2, and IntelDX4 processor on-chip floating point unit is exactly the same as the Intel387 math coprocessor. Software written for the Intel387 math coprocessor will run on the on-chip floating point unit (FPU) without any modifications.

4.2.4.1 Floating Point Data Registers

Floating point computations use the Military Intel486 DX, IntelDX2, and IntelDX4 processor FPU data registers. These eight 80-bit registers provide the equivalent capacity of twenty 32-bit registers. Each of the eight data registers is divided into ''fields'' corresponding to the FPU's extended-precision data type.

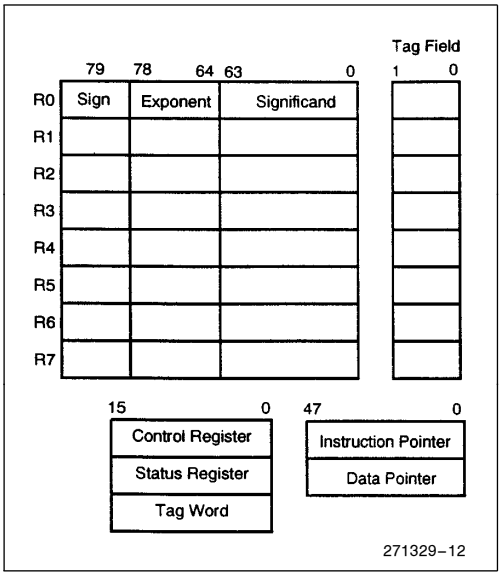


Figure 4-8. Floating Point Registers

The FPU's register set can be accessed either as a stack, with instructions operating on the top one or two stack elements, or as a fixed register set, with instructions operating on explicitly designated registers. The TOP field in the status word identifies the current top-of-stack register. A ''push'' operation decrements TOP by one and loads a value into the new top register. A ''pop'' operation stores the value from the current top register and then increments TOP by one. Like other Military Intel486 DX, IntelDX2, and IntelDX4 processor stacks in memory, the FPU register stack grows ''down'' toward lower-addressed registers.

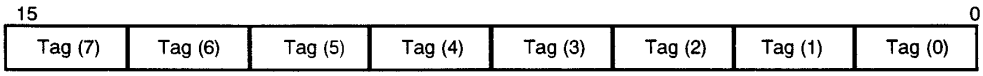
Instructions may address the data registers either implicitly or explicitly. Many instructions operate on the register at the TOP of the stack. These instructions implicitly address the register at which TOP points. Other instructions allow the programmer to explicitly specify which register to use. This explicit register addressing is also relative to TOP.

4.2.4.2 Floating Point Tag Word

The tag word marks the content of each numeric data register, as shown in Figure 4-9. Each two-bit tag represents one of the eight data registers. The principal function of the tag word is to optimize the FPU's performance and stack handling by making it possible to distinguish between empty and non-empty register locations. It also enables exception handlers to check the contents of a stack location without the need to perform complex decoding of the actual data.

4.2.4.3 Floating Point Status Word

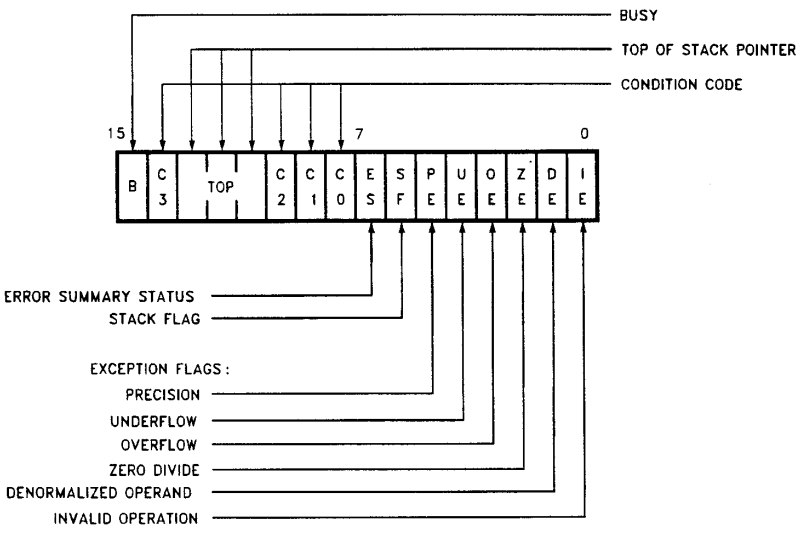
The 16-bit status word reflects the overall state of the FPU. The status word is shown in Figure 4-10 and is located in the status register.



NOTE:
The index i of tag(i) is not top-relative. A program typically uses the ''top'' field of Status Word to determine which tag(i) field refers to logical top of stack.

TAG VALUES:
00 = Valid
01 = Zero
10 = QNaN, SNaN, Infinity, Denormal and Unsupported Formats
11 = Empty

Figure 4-9. Floating Point Tag Word



271329–14

ES is set if any unmasked exception bit is set; cleared otherwise.
See Table 4-7 for interpretation of condition code.
TOP values:

000 = Register 0 is Top of Stack
001 = Register 1 is Top of Stack
•
•
•
111 = Register 7 is Top of Stack

For definitions of exceptions, refer to the section entitled ''Exception Handling''.

NOTES:
The B bit (Busy, bit 15) is included for 8087 compatibility. The B bit reflects the contents of the ES bit (bit 7 of the status word).
Bits 13–11 (TOP) point to the FPU register that is the current top-of-stack.
The four numeric condition code bits, C0–C3, are similar to the flags in EFLAGS. Instructions that perform arithmetic operations update C0–C3 to reflect the outcome. The effects of these instructions on the condition codes are summarized in Table 4-6 through Table 4-9.

Figure 4-10. Floating Point Status Word

Table 4-6. Floating Point Condition Code Interpretation

Instruction	C0 (S)	C3 (Z)	C1 (A)	C2 (C)
FPREM, FPREM1	Three least significant bits of quotient (See Table 4-8.)			Reduction 0 = complete 1 = incomplete
	Q2	Q0	Q1 or O/U#	
FCOM, FCOMP, FCOMPP, FTST, FUCOM, FUCOMP, FUCOMPP, FICOM, FICOMP	Result of comparison (see Table 4-9)		Zero or O/U#	Operand is not comparable
FXAM	Operand class (see Table 4-10)		Sign or O/U#	Operand class
FCHS, FABS, FXCH, FINCTOP, FDECTOP, Constant loads, FXTRACT, FLD, FILD, FBLD, FSTP (ext real)	UNDEFINED		Zero or O/U#	UNDEFINED
FIST, FBSTP, FRNDINT, FST, FSTP, FADD, FMUL, FDIV, FDIVR, FSUB, FSUBR, FSCALE, FSQRT, FPATAN, F2XM1, FYL2X, FYL2XP1	UNDEFINED		Roundup or O/U#	UNDEFINED
FPTAN, FSIN, FCOS, FSINCOS	UNDEFINED		Roundup or O/U#, if C2 = 1	Reduction 0 = complete 1 = incomplete
FLDENV, FRSTOR	Each bit loaded from memory			
FINIT	Clears these bits			
FLDCW, FSTENV, FSTCW, FSTSW, FCLEX, FSAVE	UNDEFINED			

NOTES:

O/U# When both IE and SF bits of status word are set, indicating a stack exception, this bit distinguishes between stack overflow (C1 = 1) and underflow (C1 = 0).

Reduction If FPREM or FPREM1 produces a remainder that is less than the modulus, reduction is complete. When reduction is incomplete the value at the top of the stack is a partial remainder, which can be used as input to further reduction. For FPTAN, FSIN, FCOS, and FSINCOS, the reduction bit is set if the operand at the top of the stack is too large. In this case the original operand remains at the top of the stack.

Roundup When the PE bit of the status word is set, this bit indicates whether the last rounding in the instruction was upward.

UNDEFINED Do not rely on finding any specific value in these bits. (See Section 4.2.7, ''Compatibility.'')

Table 4-7. Condition Code Interpretation after FPREM and FPREM1 Instructions

Condition Code				Interpretation after FPREM and FPREM1	
C2	C3	C1	C0		
1	X	X	X	Incomplete Reduction: further interaction required for complete reduction	
	Q1	Q0	Q2	Q MOD8	
0	0	0	0	0	Complete Reduction: C0, C3, and C1 contain the three least-significant bits of the quotient
	0	1	0	1	
	1	0	0	2	
	1	1	0	3	
	0	0	1	4	
	0	1	1	5	
	1	0	1	6	
	1	1	1	7	

Table 4-8. Condition Code Resulting from Comparison

Order	C3	C2	C0
TOP > Operand	0	0	0
TOP < Operand	0	0	1
TOP = Operand	1	0	0
Unordered	1	1	1

Table 4-9. Condition Code Defining Operand Class

C3	C2	C1	C0	Value at TOP
0	0	0	0	+ Unsupported
0	0	0	1	+ NaN
0	0	1	0	− Unsupported
0	0	1	1	− NaN
0	1	0	0	+ Normal
0	1	0	1	+ Infinity
0	1	1	0	− Normal
0	1	1	1	− Infinity
1	0	0	0	+ 0
1	0	0	1	+ Empty
1	0	1	0	− 0
1	0	1	1	− Empty
1	1	0	0	+ Denormal
1	1	1	0	− Denormal

Bit 7 is the error summary (ES) status bit. The ES bit is set if any unmasked exception bit (bits 0–5 in the status word) is set; ES is clear otherwise. The FERR# (floating point error) signal is asserted when ES is set.

Bit 6 is the stack flag (SF). This bit is used to distinguish invalid operations due to stack overflow or underflow. When SF is set, bit 9 (C1) distinguishes between stack overflow (C1 = 1) and underflow (C1 = 0).

Table 4-10 shows the six exception flags in bits 0–5 of the status word. Bits 0–5 are set to indicate that the FPU has detected an exception while executing an instruction.

The six exception flags in the status word can be individually masked by mask bits in the FPU control word. Table 4-10 lists the exception conditions, and their causes in order of precedence. Table 4-10 also shows the action taken by the FPU if the corresponding exception flag is masked.

An exception that is not masked by the control word will cause three things to happen: the corresponding exception flag in the status word will be set, the ES bit in the status word will be set and the FERR# output signal will be asserted. When the Military Intel486 DX, IntelDX2, or IntelDX4 processor attempts to execute another floating point or WAIT instruction, exception 16 occurs or an external interrupt happens if the NE = 1 in control register 0. The exception condition must be resolved via an interrupt service routine. The FPU saves the address of the floating point instruction that caused the exception and the address of any memory operand required by that instruction in the instruction and data pointers. (See section 4.2.4.4, ''Instruction and Data Pointers.'')

Note that when a new value is loaded into the status word by the FLDENV (load environment) or FRSTOR (restore state) instruction, the value of ES (bit 7) and its reflection in the B bit (bit 15) are not derived from the values loaded from memory. The values of ES and B are dependent upon the values of the exception flags in the status word and their corresponding masks in the control word. If ES is set in such a case, the FERR# output of the Military Intel486 DX, IntelDX2, or IntelDX4 processor is activated immediately.

Table 4-10. FPU Exceptions

Exception	Cause	Default Action (if exception is masked)
Invalid Operation	Operation on a signaling NaN, unsupported format, indeterminate form (0 * ∞, 0/0, (+ ∞) + (− ∞), etc.), or stack overflow/underflow (SF is also set).	Result is a quiet NaN, integer indefinite, or BCD indefinite
Denormalized Operand	At least one of the operands is denormalized, i.e., it has the smallest exponent but a non-zero significand.	Normal processing continues
Zero Divisor	The divisor is zero while the dividend is a non-infinite, non-zero number.	Result is ∞
Overflow	The result is too large in magnitude to fit in the specified format.	Result is largest finite value or ∞
Underflow	The true result is non-zero but too small to be represented in the specified format, and, if underflow exception is masked, denormalization causes loss of accuracy.	Result is denormalized or zero
Inexact Result (Precision)	The true result is not exactly representable in the specified format (e.g., 1/3); the result is rounded according to the rounding mode.	Normal processing continues

4.2.4.4 Instruction and Data Pointers

Because the FPU operates in parallel with the ALU (in the Military Intel486 DX, IntelDX2 and IntelDX4 processors the arithmetic and logic unit (ALU) consists of the base architecture registers), any errors detected by the FPU may be reported after the ALU has executed the floating point instruction that caused it. To allow identification of the failing numeric instruction, the Military Intel486 DX, IntelDX2, and IntelDX4 processors contain two pointer registers that supply the address of the failing numeric instruction and the address of its numeric memory operand (if appropriate).

The instruction and data pointers are provided for user-written error handlers. These registers are accessed by the FLDENV (load environment), FSTENV (store environment), FSAVE (save state) and FRSTOR (restore state) instructions. Whenever the Military Intel486 DX, IntelDX2, and IntelDX4 processors decode a new floating point instruction, it saves the instruction (including any prefixes that may be present), the address of the operand (if present) and the opcode.

The instruction and data pointers appear in one of four formats depending on the operating mode of the Military Intel486 DX, IntelDX2, and IntelDX4 processors (protected mode or real-address mode) and depending on the operand-size attribute in effect (32-bit operand or 16-bit operand). When the Military Intel486 DX, IntelDX2, or IntelDX4 processor is in the virtual-86 mode, the real address mode formats are used. The four formats are shown in Figure 4-11 through Figure 4-14. The floating point instructions FLDENV, FSTENV, FSAVE and FRSTOR are used to transfer these values to and from memory. Note that the value of the data pointer is undefined if the prior floating point instruction did not have a memory operand.

NOTE:
The operand size attribute is the D bit in a segment descriptor.

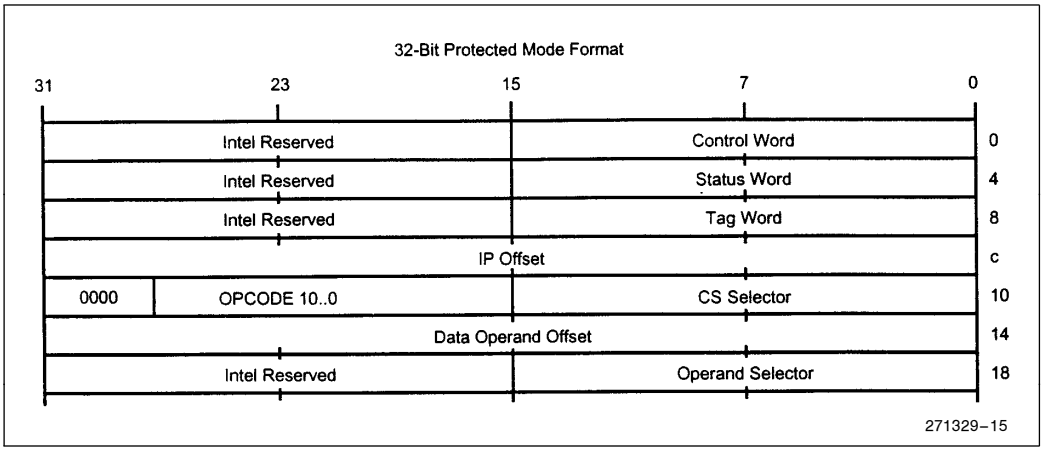


Figure 4-11. Protected Mode FPU Instructions and Data Pointer Image in Memory, 32-Bit Format

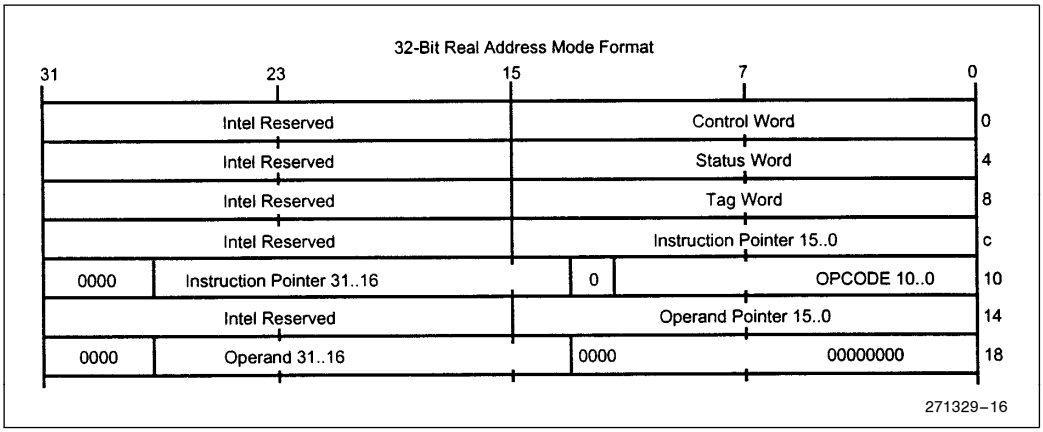


Figure 4-12. Real Mode FPU Instruction and Data Pointer Image in Memory, 32-Bit Format

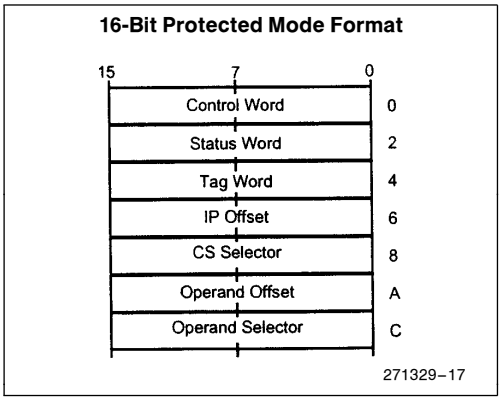


Figure 4-13. Protected Mode FPU Instruction and Data Pointer Image in Memory, 16-Bit Format

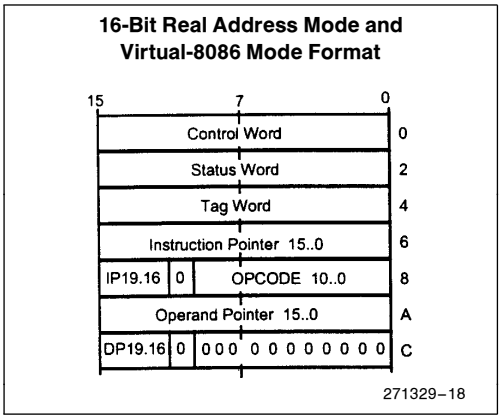
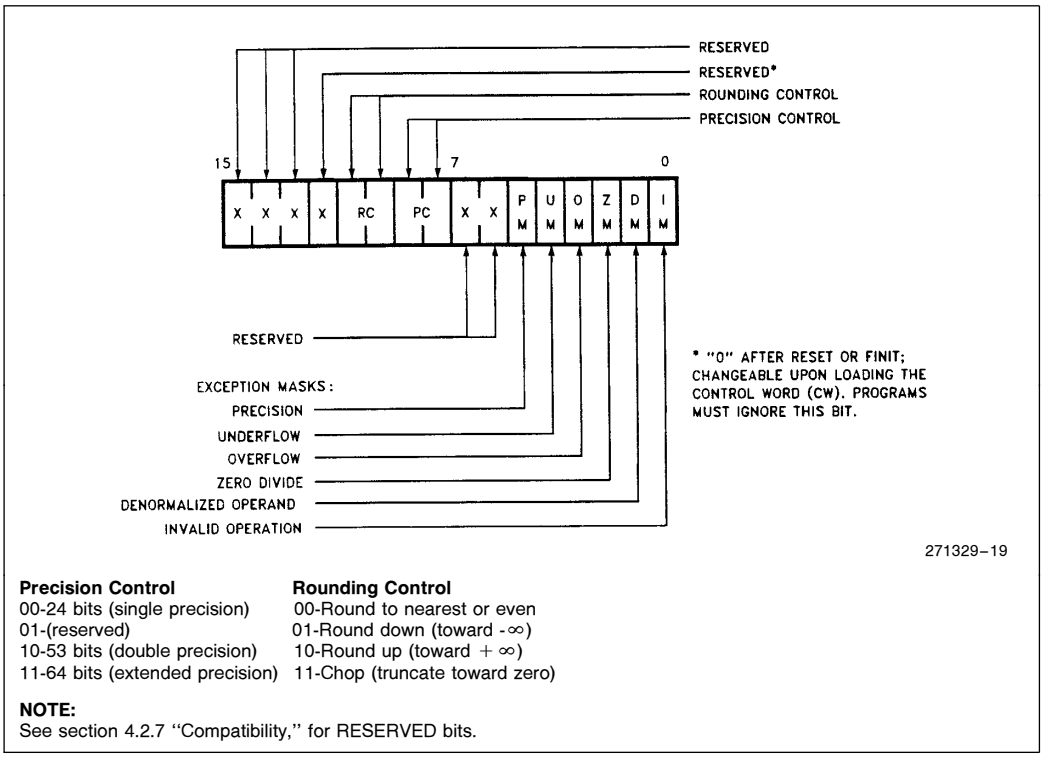


Figure 4-14. Real Mode FPU Instruction and Data Pointer Image in Memory, 16-Bit Format



Precision Control	**Rounding Control**
00-24 bits (single precision)	00-Round to nearest or even
01-(reserved)	01-Round down (toward -∞)
10-53 bits (double precision)	10-Round up (toward + ∞)
11-64 bits (extended precision)	11-Chop (truncate toward zero)

NOTE:
See section 4.2.7 "Compatibility," for RESERVED bits.

Figure 4-15. FPU Control Word

4.2.4.5 FPU Control Word

The FPU provides several processing options that are selected by loading a control word from memory into the control register. Figure 4-15 shows the format and encoding of fields in the control word.

The low-order byte of the FPU control word configures the FPU error and exception masking. Bits 0–5 of the control word contain individual masks for each of the six exceptions that the FPU recognizes.

The high-order byte of the control word configures the FPU operating mode, including precision and rounding.

RC (Rounding Control, bits 10–11)

RC bits provide for directed rounding and true chop, as well as the unbiased round to nearest even mode specified in the IEEE standard. Rounding control affects only those instructions that perform rounding at the end of the operation (and thus can generate a precision exception); namely, FST, FSTP, FIST, all arithmetic instructions (except FPREM, FPREM1, FXTRACT, FABS and FCHS), and all transcendental instructions.

PC (Precision Control, bits 8–9)

PC bits can be used to set the FPU internal operating precision of the significand at less than the default of 64 bits (extended precision). This can be useful in providing compatibility with early generation arithmetic processors of smaller precision. PC affects only the instructions ADD, SUB, DIV, MUL, and SQRT. For all other instructions, either the precision is determined by the opcode or extended precision is used.

4.2.5 DEBUG AND TEST REGISTERS

4.2.5.1 Debug Registers

The six programmer accessible debug registers (Figure 4-16) provide on-chip support for debugging. Debug registers DR0–3 specify the four linear breakpoints. The Debug control register DR7, is used to set the breakpoints and the Debug Status Register, DR6, displays the current state of the breakpoints. The use of the Debug registers is described in section 12, ''Debugging Support.''

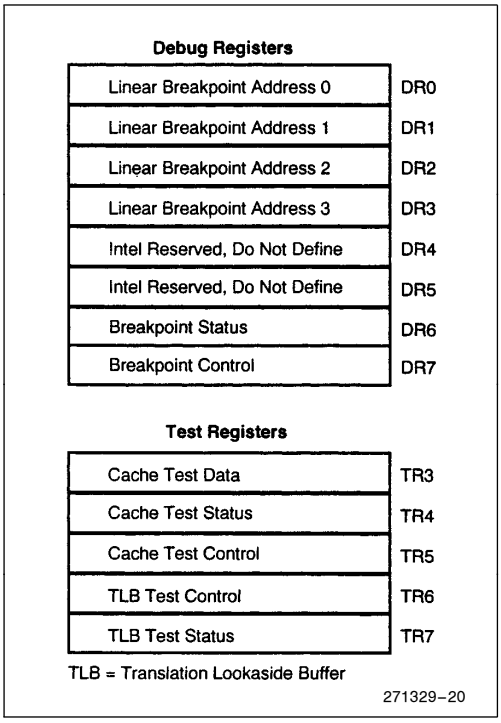


Figure 4-16. Debug and Test Registers

4.2.5.2 Test Registers

The Military Intel486 processor contains five test registers. The test registers are shown in Figure 4-16. TR6 and TR7 are used to control the testing of the translation look- aside buffer. TR3, TR4 and TR5 are used for testing the on-chip cache. The use of the test registers is discussed in section 11, ''Testability.''

4.2.6 REGISTER ACCESSIBILITY

There are a few differences regarding the accessibility of the registers in Real and Protected Mode. Table 4-11 summarizes these differences. (See section 6, ''Protected Mode Architecture.'')

4.2.6.1 FPU Register Usage

In addition to the differences listed in Table 4-11, Table 4-12 summarizes the differences for the on-chip FPU.

Table 4-11. Register Usage

Register	Use in Real Mode		Use in Protected Mode		Use in Virtual 8086 Mode	
	Load	Store	Load	Store	Load	Store
General Registers	Yes	Yes	Yes	Yes	Yes	Yes
Segment Register	Yes	Yes	Yes	Yes	Yes	Yes
Flag Register	Yes	Yes	Yes	Yes	IOPL(1)	IOPL
Control Registers	Yes	Yes	PL = 0(2)	PL = 0	No	Yes
GDTR	Yes	Yes	PL = 0	Yes	No	Yes
IDTR	Yes	Yes	PL = 0	Yes	No	Yes
LDTR	No	No	PL = 0	Yes	No	No
TR	No	No	PL = 0	Yes	No	No
Debug Registers	Yes	Yes	PL = 0	PL = 0	No	No
Test Registers	Yes	Yes	PL = 0	PL = 0	No	No

NOTES:
1. IOPL: The PUSHF and POPF instructions are made I/O Privilege Level sensitive in Virtual 8086 Mode.
2. PL = 0: The registers can be accessed only when the current privilege level is zero.

Table 4-12. FPU Register Usage Differences

Register	Use in Real Mode		Use in Protected Mode		Use in Virtual 8086 Mode	
	Load	Store	Load	Store	Load	Store
FPU Data Registers	Yes	Yes	Yes	Yes	Yes	Yes
FPU Control Registers	Yes	Yes	Yes	Yes	Yes	Yes
FPU Status Registers	Yes	Yes	Yes	Yes	Yes	Yes
FPU Instruction Pointer	Yes	Yes	Yes	Yes	Yes	Yes
FPU Data Pointer	Yes	Yes	Yes	Yes	Yes	Yes

4.2.7 COMPATIBILITY

VERY IMPORTANT NOTE:
COMPATIBILITY WITH FUTURE PROCESSORS

In the preceding register descriptions, note certain Military Intel486 processor register bits are Intel reserved. When reserved bits are called out, treat them as fully undefined. This is essential for your software compatibility with future processors! Follow the guidelines below:

1. **Do not depend on the states of any undefined bits when testing the values of defined register bits. Mask them out when testing.**
2. **Do not depend on the states of any undefined bits when storing them to memory or another register.**
3. **Do not depend on the ability to retain information written into any undefined bits.**
4. **When loading registers, always load the undefined bits as zeros.**
5. **However, registers that have been previously stored may be reloaded without masking.**

Depending upon the values of undefined register bits will make your software dependent upon the unspecified Military Intel486 processor handling of these bits. Depending on undefined values risks making your software incompatible with future processors that define usages for the Military Intel486 processor-undefined bits. AVOID ANY SOFTWARE DEPENDENCE UPON THE STATE OF UNDEFINED MILITARY INTEL486 PROCESSOR REGISTER BITS.

4.3 Instruction Set

The Military Intel486 processor instruction set can be divided into the following categories of operations:

- Data Transfer
- Arithmetic
- Shift/Rotate
- String Manipulation
- Bit Manipulation
- Control Transfer
- High Level Language Support
- Operating System Support
- Processor Control

The Military Intel486 processor instructions are listed in section 13, ''Instruction Set Summary.''

All Military Intel486 processor instructions operate on either 0, 1, 2 or 3 operands; where an operand resides in a register, in the instruction itself or in memory. Most zero operand instructions (e.g., CLI, STI) take only one byte. One operand instructions generally are two bytes long. The average instruction is 3.2-bytes long. Because the Military Intel486 processor has a 32-byte instruction queue, an average of 10 instructions will be prefetched. The use of two operands permits the following types of common instructions:

- Register to Register
- Memory to Register
- Memory to Memory
- Immediate to Register
- Register to Memory
- Immediate to Memory

The operands can be either 8-, 16-, or 32-bits long. As a general rule, when executing 32-bit code, operands are 8 or 32 bits; when executing existing 80286 or 8086 processor code (16-bit code), operands are 8 or 16 bits. Prefixes can be added to all instructions that override the default length of the operands (i.e., use 32-bit operands for 16-bit code, or 16-bit operands for 32-bit code).

4.3.1 FLOATING POINT INSTRUCTIONS

In addition to the instructions listed above, the Military Intel486, IntelDX2, and IntelDX4 processors have the following floating point instructions. Note that all floating point unit instruction mnemonics begin with an F.

- Floating Point
- Floating Point Control

4.4 Memory Organization

Memory on the Military Intel486 processor is divided up into 8-bit quantities (bytes), 16-bit quantities (words), and 32-bit quantities (dwords). Words are stored in two consecutive bytes in memory with the low-order byte at the lowest address, the high order byte at the high address. Dwords are stored in four consecutive bytes in memory with the low-order byte at the lowest address, the high-order byte at the highest address. The address of a word or dword is the byte address of the low-order byte.

In addition to these basic data types, the Military Intel486 processor supports two larger units of memory: pages and segments. Memory can be divided up into one or more variable-length segments, which can be swapped to disk or shared between programs. Memory can also be organized into one or more 4-Kbyte pages. Both segmentation and paging can be combined, gaining the advantages of both systems. The Military Intel486 processor supports both pages and segments in order to provide maximum flexibility to the system designer. Segmentation and paging are complementary. Segmentation is useful for organizing memory in logical modules, and as such is a tool for the application programmer, while pages are useful for the system programmer for managing the physical memory of a system.

4.4.1 ADDRESS SPACES

The Military Intel486 processor has three distinct address spaces: **logical, linear,** and **physical**. A **logical** address (also known as a **virtual** address) consists of a selector and an offset. A selector is the contents of a segment register. An offset is formed by summing all of the addressing components (BASE, INDEX, DISPLACEMENT) discussed in section 4.6.3 "32-Bit Memory Addressing Modes," into an effective address. Because each task on the Military Intel486 processor has a maximum of 16K ($2^{14}-1$) selectors, and offsets can be 4 Gbytes (2^{32} bits), this gives a total of 2^{46} bits or 64 terabytes of **logical** address space per task. The programmer sees this virtual address space.

The segmentation unit translates the **logical** address space into a 32-bit **linear** address space. If the paging unit is not enabled then the 32-bit **linear** address corresponds to the **physical** address. The paging unit translates the **linear** address space into the **physical** address space. The **physical address** is what appears on the address pins.

The primary difference between Real Mode and Protected Mode is how the segmentation unit performs the translation of the **logical** address into the **linear** address. In Real Mode, the segmentation unit shifts the selector left four bits and adds the result to the offset to form the **linear** address. While in Protected Mode every selector has a **linear** base address associated with it. The **linear base** address is stored in one of two operating system tables (i.e., the Local Descriptor Table or Global Descriptor Table). The selector's **linear base** address is added to the offset to form the final **linear** address.

Figure 4-17 shows the relationship between the various address spaces.

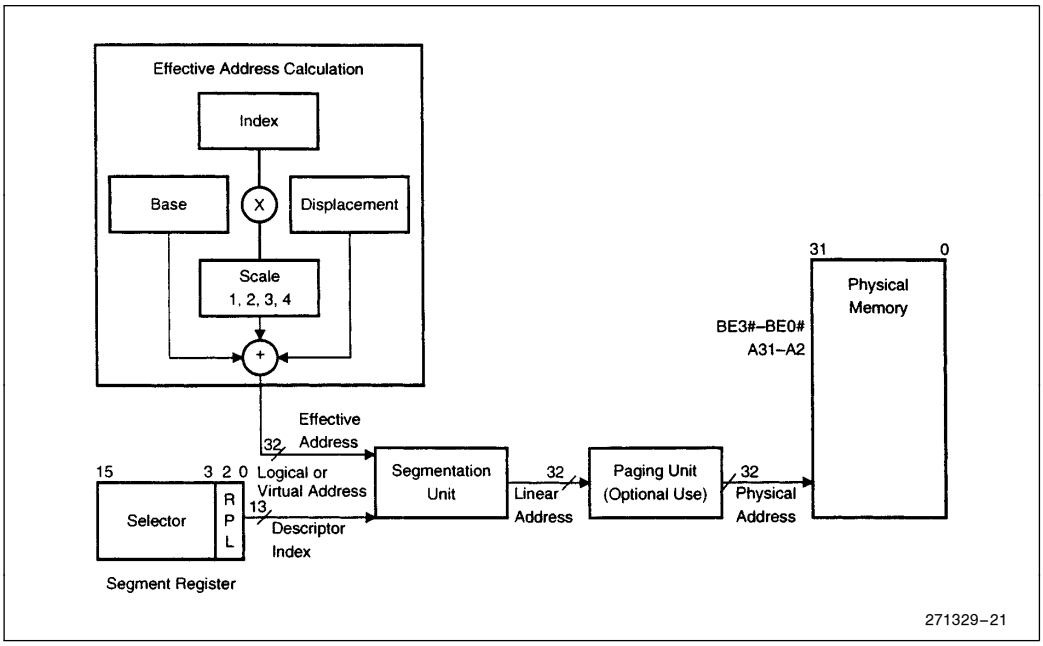


Figure 4-17. Address Translation

4.4.2 SEGMENT REGISTER USAGE

The main data structure used to organize memory is the segment. On the Military Intel486 processor, segments are variable sized blocks of linear addresses which have certain attributes associated with them. There are two main types of segments: code and data. The segments are of variable size and can be as small as 1 byte or as large as 4 Gbytes (2^{32} bytes).

In order to provide compact instruction encoding, and increase Military Intel486 processor performance, instructions do not need to explicitly specify which segment register is used. A default segment register is automatically chosen according to the rules of Table 4-13. In general, data references use the selector contained in the DS register; Stack references use the SS register and Instruction fetches use the CS register. The contents of the Instruction Pointer provide the offset. Special segment override prefixes allow the explicit use of a given segment register, and override the implicit rules listed in Table 4-13. The override prefixes also allow the use of the ES, FS and GS segment registers.

There are no restrictions regarding the overlapping of the base addresses of any segments. Thus, all 6 segments could have the base address set to zero and create a system with a 4-Gbyte linear address space. This creates a system where the virtual address space is the same as the linear address space. Further details of segmentation are discussed in section 6.0, ''Protected Mode Architecture.''

4.5 I/O Space

The Military Intel486 processor has two distinct physical address spaces: Memory and I/O. Generally, peripherals are placed in I/O space although the Military Intel486 processor also supports memory-mapped peripherals. The I/O space consists of 64 Kbytes, it can be divided into 64K 8-bit ports, 32K 16-bit ports, or 16K 32-bit ports, or any combination of ports which add up to less than 64 Kbytes. The 64K I/O address space refers to physical memory rather than linear address, because I/O instructions do not go through the segmentation or paging hardware. The M/IO# pin acts as an additional address line thus allowing the system designer to easily determine which address space the processor is accessing.

Table 4-13. Segment Register Selection Rules

Type of Memory Reference	Implied (Default) Segment Use	Segment Override Prefixes Possible
Code Fetch	CS	None
Destination of PUSH, PUSHF, INT, CALL, PUSHA Instructions	SS	None
Source of POP, POPA, POPF, IRET, RET instructions	SS	None
Destination of STOS, MOVS, REP STOS, REP MOVS Instructions (DI is Base Register)	ES	None
Other Data References, with Effective Address using Base Register of:		
[EAX]	DS	All
[EBX]	DS	
[ECX]	DS	
[EDX]	DS	
[ESI]	DS	
[EDI]	DS	
[EBP]	SS	
[ESP]	SS	

The I/O ports are accessed via the IN and OUT I/O instructions, with the port address supplied as an immediate 8-bit constant in the instruction or in the DX register. All 8- and 16-bit port addresses are zero extended on the upper address lines. The I/O instructions cause the M/IO# pin to be driven low.

I/O port addresses 00F8H through 00FFH are reserved for use by Intel.

I/O instruction code is cacheable.

I/O data is not cacheable.

I/O transfers (data or code) can be bursted.

4.6 Addressing Modes

4.6.1 ADDRESSING MODES OVERVIEW

The Military Intel486 processor provides a total of 11 addressing modes for instructions to specify operands. The addressing modes are optimized to allow the efficient execution of high-level languages such as C and FORTRAN, and they cover the vast majority of data references needed by high-level languages.

4.6.2 REGISTER AND IMMEDIATE MODES

The following two addressing modes provide for instructions that operate on register or immediate operands:

- **Register Operand Mode:** The operand is located in one of the 8-, 16- or 32-bit general registers.
- **Immediate Operand Mode:** The operand is included in the instruction as part of the opcode.

4.6.3 32-BIT MEMORY ADDRESSING MODES

The remaining modes provide a mechanism for specifying the effective address of an operand. The linear address consists of two components: the segment base address and an effective address. The effective address is calculated by using combinations of the following four address elements:

- **DISPLACEMENT:** An 8-, or 32-bit immediate value, following the instruction.
- **BASE:** The contents of any general purpose register. The base registers are generally used by compilers to point to the start of the local variable area.
- **INDEX:** The contents of any general purpose register except for ESP. The index registers are used to access the elements of an array, or a string of characters.
- **SCALE:** The index register's value can be multiplied by a scale factor, either 1, 2, 4 or 8. Scaled index mode is especially useful for accessing arrays or structures.

Combinations of these 4 components make up the 9 additional addressing modes. There is no performance penalty for using any of these addressing combinations, because the effective address calculation is pipelined with the execution of other instructions. The one exception is the simultaneous use of Base and Index components, which requires one additional clock.

As shown in Figure 4-18, the effective address (EA) of an operand is calculated according to the following formula:

EA = Base Reg + (Index Reg * Scaling) + Displacement

Direct Mode: The operand's offset is contained as part of the instruction as an 8-, 16- or 32-bit displacement.

Example: INC Word PTR [500]

Register Indirect Mode: A BASE register contains the address of the operand.

Example: MOV [ECX], EDX

Based Mode: A BASE register's contents is added to a DISPLACEMENT to form the operand's offset.

Example: MOV ECX, [EAX+24]

Index Mode: An INDEX register's contents is added to a DISPLACEMENT to form the operand's offset.

Example: ADD EAX, TABLE[ESI]

Scaled Index Mode: An INDEX register's contents is multiplied by a scaling factor which is added to a DISPLACEMENT to form the operand's offset.

Example: IMUL EBX, TABLE[ESI*4],7

Based Index Mode: The contents of a BASE register is added to the contents of an INDEX register to form the effective address of an operand.

Example: MOV EAX, [ESI] [EBX]

Based Scaled Index Mode: The contents of an INDEX register is multiplied by a SCALING factor and the result is added to the contents of a BASE register to obtain the operand's offset.

Example: MOV ECX, [EDX*8] [EAX]

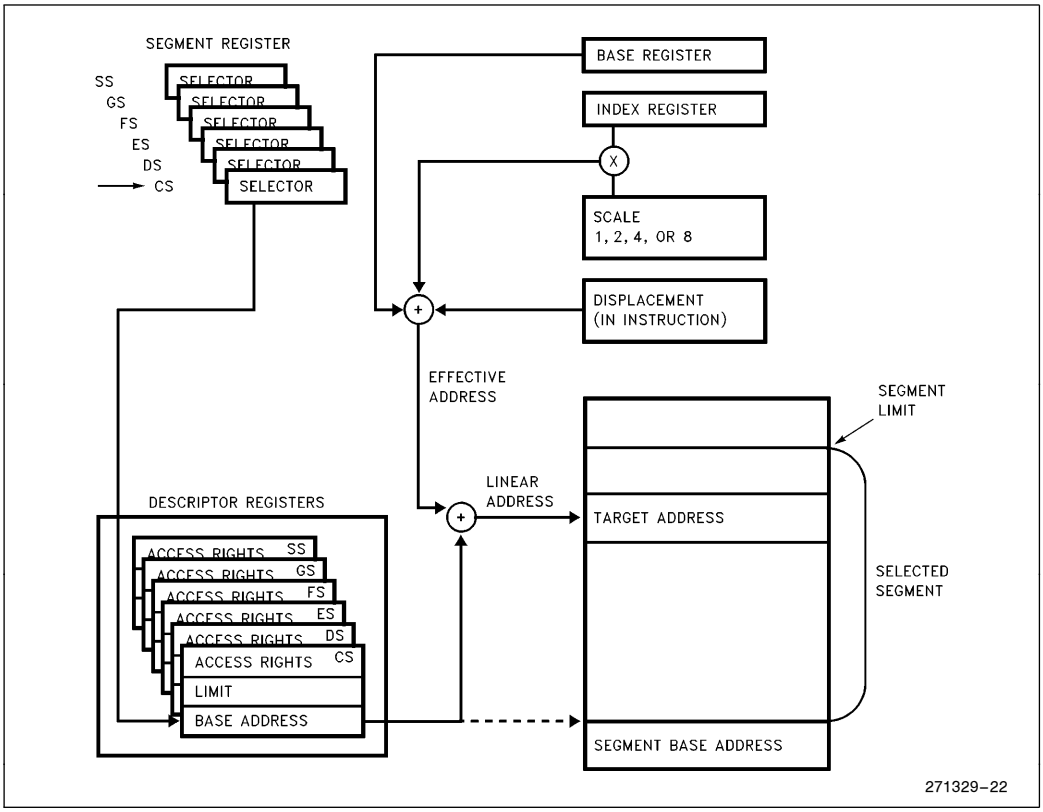


Figure 4-18. Addressing Mode Calculations

Based Index Mode with Displacement: The contents of an INDEX Register and a BASE register's contents and a DISPLACEMENT are all summed together to form the operand offset.

Example: ADD EDX, [ESI] [EBP + 00FFFFF0H]

Based Scaled Index Mode with Displacement: The contents of an INDEX register are multiplied by a SCALING factor, the result is added to the contents of a BASE register and a DISPLACEMENT to form the operand's offset.

Example: MOV EAX, LOCALTABLE[EDI*4] [EBP + 80]

4.6.4 DIFFERENCES BETWEEN 16- AND 32-BIT ADDRESSES

In order to provide software compatibility with 80286 and 8086 processors, the Military Intel486 processor can execute 16-bit instructions in Real and Protected Modes. The processor determines the size of the instructions it is executing by examining the D bit in the CS segment Descriptor. If the D bit is 0 then all operand lengths and effective addresses are assumed to be 16 bits long. If the D bit is 1 then the default length for operands and addresses is 32 bits. In Real Mode the default size for operands and addresses is 16-bits.

Regardless of the default precision of the operands or addresses, the Military Intel486 processor is able to execute either 16- or 32-bit instructions. This is

specified via the use of override prefixes. Two prefixes, the **Operand Size Prefix** and the **Address Length Prefix**, override the value of the D bit on an individual instruction basis. These prefixes are automatically added by Intel assemblers.

Example: The Military Intel486 processor is executing in Real Mode and the programmer needs to access the EAX registers. The assembler code for this might be MOV EAX, 32-bit MEMORYOP, ASM486 Macro Assembler automatically determines that an Operand Size Prefix is needed and generates it.

Example: The D bit is 0, and the programmer wishes to use Scaled Index addressing mode to access an array. The Address Length Prefix allows the use of MOV DX, TABLE[ESI*2]. The assembler uses an Address Length Prefix because, with D = 0, the default addressing mode is 16-bits.

Example: The D bit is 1, and the program wants to store a 16-bit quantity. The Operand Length Prefix is used to specify only a 16-bit value; MOV MEM16, DX.

The OPERAND LENGTH and Address Length Prefixes can be applied separately or in combination to any instruction. The Address Length Prefix does not allow addresses over 64 Kbytes to be accessed in Real Mode. A memory address which exceeds FFFFH will result in a General Protection Fault. An Address Length Prefix only allows the use of the additional Military Intel486 processor addressing modes.

When executing 32-bit code, the Military Intel486 processor uses either 8-, or 32-bit displacements, and any register can be used as base or index registers. When executing 16-bit code, the displacements are either 8, or 16 bits, and the base and index register conform to the 80286 processor model. Table 4-14 illustrates the differences.

Table 4-14. BASE and INDEX Registers for 16- and 32-Bit Addresses

	16-Bit Addressing	32-Bit Addressing
BASE REGISTER	BX,BP	Any 32-bit GP Register
INDEX REGISTER	SI,DI	Any 32-bit GP Register Except ESP
SCALE FACTOR	none	1, 2, 4, 8
DISPLACEMENT	0, 8, 16 bits	0, 8, 32 bits

4.7 Data Formats

4.7.1 DATA TYPES

The Military Intel486 processor can support a wide-variety of data types. In the following descriptions, the processor consists of the base architecture registers.

4.7.1.1 Unsigned Data Types

Byte: Unsigned 8-bit quantity

Word: Unsigned 16-bit quantity

Dword: Unsigned 32-bit quantity

The least significant bit (LSB) in a byte is bit 0, and the most significant bit is 7.

4.7.1.2 Signed Data Types

All signed data types assume 2's complement notation. The signed data types contain two fields, a sign bit and a magnitude. The sign bit is the most significant bit (MSB). The number is negative if the sign bit is 1. If the sign bit is 0, the number is positive. The magnitude field consists of the remaining bits in the number. (Refer to Figure 4-19.)

8-bit Integer: Signed 8-bit quantity

16-bit Integer: Signed 16-bit quantity

32-bit Integer: Signed 32-bit quantity

64-bit Integer: Signed 64-bit quantity

The integer core of the Military Intel486 processors only support 8-, 16- and 32-bit integers. (See section 4.7.1.4, ''Floating Point Data Types.'')

4.7.1.3 BCD Data Types

The Military Intel486 processor supports packed and unpacked binary coded decimal (BCD) data types. A packed BCD data type contains two digits per byte, the lower digit is in bits 0–3 and the upper digit in bits 4–7. An unpacked BCD data type contains 1 digit per byte stored in bits 0–3.

The Military Intel486 processor supports 8-bit packed and unpacked BCD data types. (Refer to Figure 4-19.)

4.7.1.4 Floating Point Data Types

In addition to the base registers, the Military Intel486 DX, IntelDX2, and IntelDX4 processors' on-chip floating point unit consists of the floating point registers. The floating point unit data type contain three fields: sign, significand and exponent. The sign field is one bit and is the MSB of the floating point number. The number is negative if the sign bit is 1. If the sign bit is 0, the number is positive. The significand gives the significant bits of the number. The exponent field contains the power of 2 needed to scale the significand. (Refer to Figure 4-19.)

Only the FPU supports floating point data types.

Single Precision Real:	23-bit significand and 8-bit exponent. 32 bits total.
Double Precision Real:	52-bit significand and 11-bit exponent. 64 bits total.
Extended Precision Real:	64-bit significand and 15-bit exponent. 80 bits total.

Floating Point Unsigned Data Types

The on-chip FPU does not support unsigned data types. (Refer to Figure 4-19.)

Floating Point Signed Data Types

The on-chip FPU only supports 16-, 32- and 64-bit integers.

Floating Point BCD Data Types

The on-chip FPU only supports 80-bit packed BCD data types.

4.7.1.5 String Data Types

A string data type is a contiguous sequence of bits, bytes, words or dwords. A string may contain between 1 byte and 4 Gbytes. (Refer to Figure 4-20.)

String data types are only supported by the CPU section of the Military Intel486 processor.

Byte String:	Contiguous sequence of bytes.
Word String:	Contiguous sequence of words.
Dword String:	Contiguous sequence of dwords.
Bit String:	A set of contiguous bits. In the Military Intel486 processor bit strings can be up to 4-gigabits long.

4.7.1.6 ASCII Data Types

The Military Intel486 processor supports ASCII (American Standard Code for Information Interchange) strings and can perform arithmetic operations (such as addition and division) on ASCII data. The Military Intel486 processor can only operate on ASCII data. (Refer to Figure 4-20.)

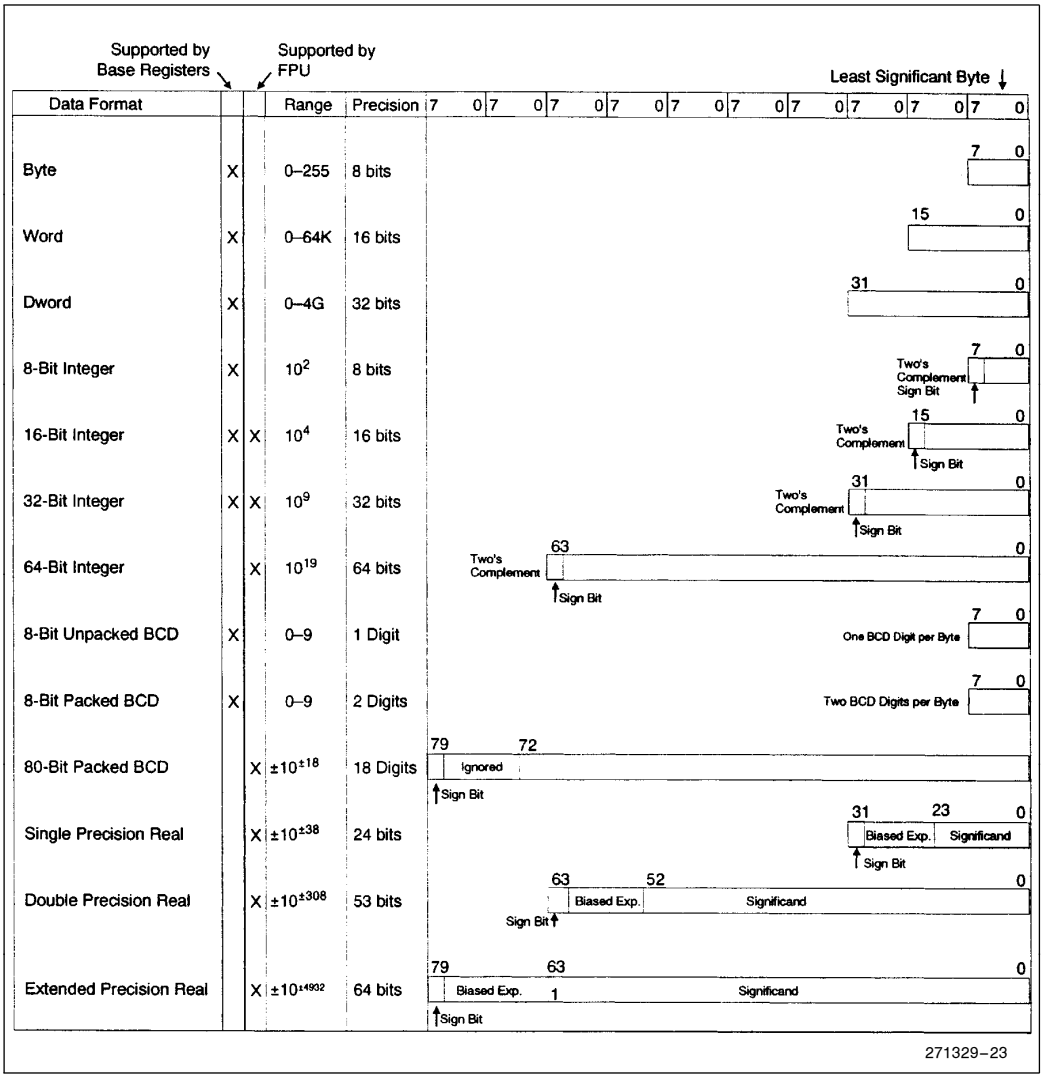


Figure 4-19. Military Intel486™ Processor Data Types

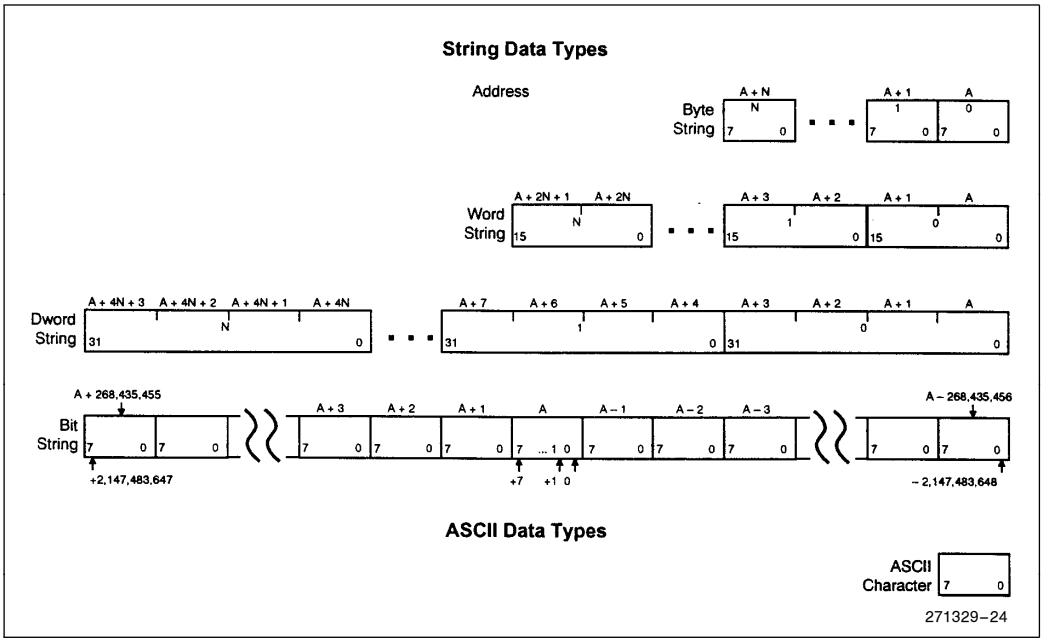


Figure 4-20. String and ASCII Data Types

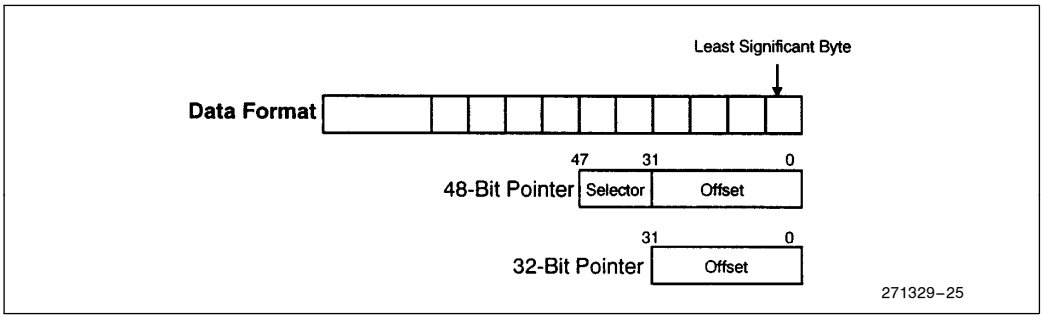


Figure 4-21. Pointer Data Types

4.7.1.7 Pointer Data Types

A pointer data type contains a value that gives the address of a piece of data. Military Intel486 processors support the following two types of pointers (see Figure 4-21):

- 48-bit Pointer: 16-bit selector and 32-bit offset
- 32-bit Pointer: 32-bit offset

4.7.2 LITTLE ENDIAN vs. BIG ENDIAN DATA FORMATS

The Military Intel486 processors, as well as all other members of the Intel architecture, use the ''little-endian'' method for storing data types that are larger than one byte. Words are stored in two consecutive bytes in memory with the low-order byte at the lowest address and the high order byte at the high address. Dwords are stored in four consecutive bytes in memory with the low-order byte at the lowest address and the high order byte at the highest address. The address of a word or dword data item is the byte address of the low-order byte.

Figure 4-22 illustrates the differences between the big-endian and little-endian formats for dwords. The 32 bits of data are shown with the low order bit numbered bit 0 and the high order bit numbered 32. Big-endian data is stored with the high-order bits at the lowest addressed byte. Little-endian data is stored with the high-order bits in the highest addressed byte.

The Military Intel486 processor has the following two instructions that can convert 16- or 32-bit data between the two byte orderings:

- BSWAP (byte swap) handles 4-byte values
- XCHG (exchange) handles 2-byte values

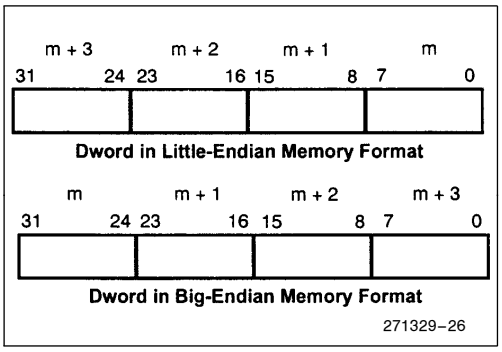


Figure 4-22. Big vs. Little Endian Memory Format

4.8 Interrupts

4.8.1 INTERRUPTS AND EXCEPTIONS

Interrupts and exceptions alter the normal program flow, in order to handle external events, to report errors or exceptional conditions. The difference between interrupts and exceptions is that interrupts are used to handle asynchronous external events while exceptions handle instruction faults. Although a program can generate a software interrupt via an INT N instruction, the Military Intel486 processors treat software interrupts as exceptions.

Hardware interrupts occur as the result of an external event and are classified into two types: maskable or non-maskable. Interrupts are serviced after the execution of the current instruction. After the interrupt handler is finished servicing the interrupt, execution proceeds with the instruction immediately **after** the interrupted instruction. Sections 4.8.3, ''Maskable Interrupt,'' and 4.8.4, ''Non-Maskable Interrupt,'' discuss the differences between Maskable and Non-Maskable interrupts.

Exceptions are classified as faults, traps, or aborts, depending on the way they are reported, and whether or not restart of the instruction causing the exception is supported. **Faults** are exceptions that are detected and serviced **before** the execution of the faulting instruction. A fault would occur in a virtual memory system when the processor referenced a page or a segment that was not present. The operating system would fetch the page or segment from disk, and then the Military Intel486 processor would restart the instruction. **Traps** are exceptions that are reported immediately **after** the execution of the instruction that caused the problem. User defined interrupts are examples of traps. **Aborts** are exceptions that do not permit the precise location of the instruction causing the exception to be determined. Aborts are used to report severe errors, such as a hardware error or illegal values in system tables.

Thus, when an interrupt service routine has been completed, execution proceeds from the instruction immediately following the interrupted instruction. On the other hand, the return address from an exception fault routine will always point at the instruction causing the exception and include any leading instruction prefixes. Tables 4-15 and 4-16 summarize the possible interrupts for Military Intel486 processors and shows where the return address points.

Military Intel486 processors can handle up to 256 different interrupts and/or exceptions. In order to service the interrupts, a table with up to 256 interrupt vectors must be defined. The interrupt vectors are simply pointers to the appropriate interrupt service routine. In Real Mode (see section 5.0, ''Real Mode Architecture''), the vectors are 4-byte quantities, a Code Segment plus a 16-bit offset; in Protected Mode, the interrupt vectors are 8-byte quantities, which are put in an Interrupt Descriptor Table. (See section 6.2.3.4, ''Interrupt Descriptor Table.'') Of the 256 possible interrupts, 32 are reserved for use by Intel, the remaining 224 are free to be used by the system designer.

4.8.2 INTERRUPT PROCESSING

When an interrupt occurs, the following actions happen. First, the current program address and the Flags are saved on the stack to allow resumption of the interrupted program. Next, an 8-bit vector is supplied to the Military Intel486 processor which identifies the appropriate entry in the interrupt table. The table contains the starting address of the interrupt service routine. Then, the user supplied interrupt service routine is executed. Finally, when an IRET instruction is executed the old Military Intel486 processor state is restored and program execution resumes at the appropriate instruction.

The 8-bit interrupt vector is supplied to the Military Intel486 processor in several different ways: exceptions supply the interrupt vector internally; software INT instructions contain or imply the vector; maskable hardware interrupts supply the 8-bit vector via the interrupt acknowledge bus sequence. Non-Maskable hardware interrupts are assigned to interrupt vector 2.

4.8.3 MASKABLE INTERRUPT

Maskable interrupts are the most common way used by the Military Intel486 processor to respond to asynchronous external hardware events. A hardware interrupt occurs when the INTR is pulled high and the Interrupt Flag bit (IF) is enabled. The Military Intel486 processor only responds to interrupts between instructions, (REPeat String instructions, have an ''interrupt window,'' between memory moves, which allows interrupts during long string moves). When an interrupt occurs, the Military Intel486 processor reads an 8-bit vector supplied by the hardware which identifies the source of the interrupt, (one of 224 user defined interrupts). The exact nature of the interrupt sequence is discussed in section 10.2.10, ''Interrupt Acknowledge.''

The IF bit in the EFLAG registers is reset when an interrupt is being serviced. This effectively disables servicing additional interrupts during an interrupt service routine. However, the IF may be set explicitly by the interrupt handler, to allow the nesting of interrupts. When an IRET instruction is executed, the original state of the IF is restored.

Table 4-15. Interrupt Vector Assignments

Function	Interrupt Number	Instruction that Can Cause Exception	Return Address Points to Faulting Instruction	Type
Divide Error	0	DIV, IDIV	YES	FAULT
Debug Exception	1	Any instruction	YES	TRAP*
NMI Interrupt	2	INT 2 or NMI	NO	NMI
One Byte Interrupt	3	INT	NO	TRAP
Interrupt on Overflow	4	INTO	NO	TRAP
Array Bounds Check	5	BOUND	YES	FAULT
Invalid OP-Code	6	Any illegal instruction	YES	FAULT
Device Not Available	7	ESC, WAIT	YES	FAULT
Double Fault	8	Any instruction that can generate an exception		ABORT
Intel Reserved	9			
Invalid TSS	10	JMP, CALL, IRET, INT	YES	FAULT
Segment Not Present	1	Segment Register Instructions	YES	FAULT
Stack Fault	12	Stack References	YES	FAULT
General Protection Fault	13	Any Memory Reference	YES	FAULT
Page Fault	14	Any Memory Access or Code Fetch	YES	FAULT
Intel Reserved	15			
Alignment Check Interrupt	17	Unaligned Memory Access	YES	FAULT
Intel Reserved	18–31			
Two Byte Interrupt	0–255	INT n	NO	TRAP

*Some debug exceptions may report both traps on the previous instruction, and faults on the next instruction.

Table 4-16. FPU Interrupt Vector Assignments

Function	Interrupt Number	Instruction Which Can Cause Exception	Return Address Points to Faulting Instruction	Type
Floating Point Error	16	Floating Point, WAIT	YES	FAULT

4.8.4 NON-MASKABLE INTERRUPT

Non-maskable interrupts provide a method of servicing very high priority interrupts. A common example of the use of a non-maskable interrupt (NMI) would be to activate a power failure routine or SMI# to activate a power saving mode. When the NMI input is pulled high, it causes an interrupt with an internally supplied vector value of 2. Unlike a normal hardware interrupt, no interrupt acknowledgment sequence is performed for an NMI.

While executing the NMI servicing procedure, the Military Intel486 processor will not service further NMI requests until an interrupt return (IRET) instruction is executed or the processor is reset (RSM in the case of SMI#). If NMI occurs while currently servicing an NMI, its presence will be saved for servicing after executing the first IRET instruction. The IF bit is cleared at the beginning of an NMI interrupt to inhibit further INTR interrupts.

4.8.5 SOFTWARE INTERRUPTS

A third type of interrupt/exception for the Military Intel486 processor is the software interrupt. An INT n instruction causes the processor to execute the interrupt service routine pointed to by the *n*th vector in the interrupt table.

A special case of the two byte software interrupt INT n is the one byte INT 3, or breakpoint interrupt. By inserting this one byte instruction in a program, the user can set breakpoints in his program as a debugging tool.

A final type of software interrupt is the single step interrupt. It is discussed in section 12.2, ''Single-Step Trap.''

4.8.6 INTERRUPT AND EXCEPTION PRIORITIES

Interrupts are externally-generated events. Maskable Interrupts (on the INTR input) and Non-Maskable Interrupts (on the NMI input or SMI# input) are recognized at instruction boundaries. When more than one interrupt or external event are **both** recognized at the **same** instruction boundary, the Military Intel486 processor invokes the highest priority routine first. (See list below.) If, after the NMI service routine has been invoked, maskable interrupts are still enabled, then the Military Intel486 processor will invoke the appropriate interrupt service routine.

Priority for Servicing External Events for All Military Intel486 Processors

1. RESET/SRESET
2. FLUSH#
3. SMI#
4. NMI
5. INTR
6. STPCLK#

NOTE:
STPCLK# will be recognized while in an interrupt service routine or an SMM handler.

Exceptions are internally-generated events. Exceptions are detected by the Military Intel486 processor if, in the course of executing an instruction, the Military Intel486 processor detects a problematic condition. The IntelDX4 processor then immediately invokes the appropriate exception service routine. The state of the Military Intel486 processor is such that the instruction causing the exception can be restarted. If the exception service routine has taken care of the problematic condition, the instruction will execute without causing the same exception.

It is possible for a single instruction to generate several exceptions (for example, transferring a single operand could generate two page faults if the operand location spans two ''not present'' pages). However, only one exception is generated upon each attempt to execute the instruction. Each exception service routine should correct its corresponding exception, and restart the instruction. In this manner, exceptions are serviced until the instruction executes successfully.

As the Military Intel486 processor executes instructions, it follows a consistent cycle in checking for exceptions. Consider the case of the Military Intel486 processor having just completed an instruction. It then performs the checks listed in Table 4-17 before reaching the point where the next instruction is completed. This cycle is repeated as each instruction is executed, and occurs in parallel with instruction decoding and execution. Checking for EM, TS, or FPU error status only occurs for processors with on-chip floating point units.

Table 4-17. Sequence of Exception Checking

Sequence	Description
1	Check for Exception 1 Traps from the instruction just completed (single-step via Trap Flag, or Data Breakpoints set in the Debug Registers).
2	Check for Exception 1 Faults in the next instruction (Instruction Execution Breakpoint set in the Debug Registers for the next instruction).
3	Check for external NMI and INTR.
4	Check for Segmentation Faults that prevented fetching the entire next instruction (exceptions 11 or 13).
5	Check for Page Faults that prevented fetching the entire next instruction (exception 14).
6	Check for Faults decoding the next instruction (exception 6 if illegal opcode; exception 6 if in Real Mode or in Virtual 8086 Mode and attempting to execute an instruction for Protected Mode only (see section 6.5.4, ''Protection and I/O Permission Bitmap''); or exception 13 if instruction is longer than 15 bytes, or privilege violation in Protected Mode (i.e., not at IOPL or at CPL = 0).
7	If WAIT opcode, check if TS = 1 and MP = 1 (exception 7 if both are 1).
8	If opcode for Floating Point Unit, check if EM = 1 or TS = 1 (exception 7 if either are 1).
9	If opcode for Floating Point Unit (FPU), check FPU error status (exception 16 if error status is asserted).
10	Check in the following order for each memory reference required by the instruction: a. Check for Segmentation Faults that prevent transferring the entire memory quantity (exceptions 11, 12, 13). b. Check for Page Faults that prevent transferring the entire memory quantity (exception 14).

NOTE:
The order stated supports the concept of the paging mechanism being ''underneath'' the segmentation mechanism. Therefore, for any given code or data reference in memory, segmentation exceptions are generated before paging exceptions are generated.

4.8.7 INSTRUCTION RESTART

The Military Intel486 processor fully supports restarting all instructions after faults. If an exception is detected in the instruction to be executed (exception categories 4 through 10 in Table 4-17), the Military Intel486 processor invokes the appropriate exception service routine.

The Military Intel486 processor is in a state that permits restart of the instruction, for all cases except the following. An instruction causes a task switch to a task whose Task State Segment is **partially** ''not present.'' (An entirely ''not present'' TSS is restartable.) Partially present TSSs can be avoided either by keeping the TSSs of such tasks present in memory, or by aligning TSS segments to reside entirely within a single 4K page (for TSS segments of 4 Kbytes or less).

NOTE:
Such cases are easily avoided by proper design of the operating system.

4.8.8 DOUBLE FAULT

A Double Fault (exception 8) results when the Military Intel486 processor attempts to invoke an exception service routine for the segment exceptions (10, 11, 12 or 13), but in the process of doing so, detects an exception other than a Page Fault (exception 14).

A Double Fault (exception 8) will also be generated when the Military Intel486 processor attempts to invoke the Page Fault (exception 14) service routine, and detects an exception other than a second Page Fault. In any functional system, the entire Page Fault service routine must remain ''present'' in memory.

When a Double Fault occurs, the Military Intel486 processor invokes the exception service routine for exception 8.

4.8.9 FLOATING POINT INTERRUPT VECTORS

Several interrupt vectors of the Military Intel486 DX, IntelDX2, and IntelDX4 processors are used to report exceptional conditions while executing numeric programs in either real or protected mode. Table 4-18 shows these interrupts and their causes.

Table 4-18. Interrupt Vectors Used by FPU

Interrupt Number	Cause of Interrupt
7	A Floating Point instruction was encountered when EM or TS of the Military Intel486 DX, IntelDX2, and IntelDX4 processor control register zero (CR0) was set. EM = 1 indicates that software emulation of the instruction is required. When TS is set, either a Floating Point or WAIT instruction causes interrupt 7. This indicates that the current FPU context may not belong to the current task.
13	The first word or doubleword of a numeric operand is not entirely within the limit of its segment. The return address pushed onto the stack of the exception handler points at the Floating Point instruction that caused the exception, including any prefixes. The FPU has not executed this instruction; the instruction pointer and data pointer register refer to a previous, correctly executed instruction.
16	The previous numerics instruction caused an unmasked exception. The address of the faulty instruction and the address of its operand are stored in the instruction pointer and data pointer registers. Only Floating Point and WAIT instructions can cause this interrupt. The Military Intel486 DX, IntelDX2, and IntelDX4 processors return address pushed onto the stack of the exception handler points to a WAIT or Floating Point instruction (including prefixes). This instruction can be restarted after clearing the exception condition in the FPU. The FNINIT, FNCLEX, FNSTSW, FNSTENV, and FNSAVE instructions can not cause this interrupt.

5.0 REAL MODE ARCHITECTURE

5.1 Introduction

When the Military Intel486 processor is reset or powered up, it is initialized in Real Mode. Real Mode has the same base architecture as the 8086 processor, except that it allows access to the 32-bit register set of the Military Intel486 processor. The Military Intel486 processor addressing mechanism, memory size and interrupt handling are identical to those of Real Mode on the 80286 processor.

All of the Military Intel486 processor instructions are available in Real Mode (except those instructions listed in section 6.5.4, ''Protection and I/O Permission Bitmap''). The default operand size in Real Mode is 16 bits, as in the 8086 processor. In order to use the 32-bit registers and addressing modes, override prefixes must be used. Also, the segment size on the Military Intel486 processor in Real Mode is 64 Kbytes, forcing 32-bit effective addresses to have a value less than 0000FFFFH. The primary purpose of Real Mode is to enable Protected Mode Operation.

The LOCK prefix on the Military Intel486 processor, even in Real Mode, is more restrictive than on the 80286 processor. This is due to the addition of paging on the Military Intel486 processor in Protected Mode and Virtual 8086 Mode. Paging makes it impossible to guarantee that repeated string instructions can be LOCKed. The Military Intel486 processor can not require that all pages holding the string be physically present in memory. Hence, a Page Fault (exception 14) might have to be taken during the repeated string instruction. Therefore, the LOCK prefix can not be supported during repeated string instructions.

Table 5-1 lists the only instruction forms where the LOCK prefix is legal on the Military Intel486 processor.

An exception 6 will be generated if a LOCK prefix is placed before any instruction form or opcode not listed above. The LOCK prefix allows indivisible read/modify/write operations on memory operands using the instructions above. For example, even the ADD Reg, Mem is not LOCKable, because the Mem operand is not the destination (and therefore no memory read/modify/operation is being performed).

Because, on the Military Intel486 processor, repeated string instructions are not LOCKable, it is not possible to LOCK the bus for a long period of time. Therefore, the LOCK prefix is not IOPL-sensitive on the Military Intel486 processor. The LOCK prefix can be used at any privilege level, but only on the instruction forms listed above.

Table 5-1. Instruction Forms Where LOCK Prefix Is Legal

Opcode	Operands (Dest, Source)
BIT Test and SET/RESET/COMPLEMENT	Mem, Reg/immed
XCHG	Reg, Mem
CHG	Mem, Reg
ADD, OR, ADC, SBB, AND, SUB, XOR	Mem, Reg/immed
NOT, NEG, INC, DEC	Mem
CMPXCHG, XADD	Mem, Reg

5.2 Memory Addressing

In Real Mode the maximum memory size is limited to 1 megabyte. (See Figure 5-1.) Thus, only address lines A2–A19 are active. (Exception, after RESET address lines A20–A31 are high during CS-relative memory cycles until an intersegment jump or call is executed. See section 9.5, ''Reset and Initialization''.)

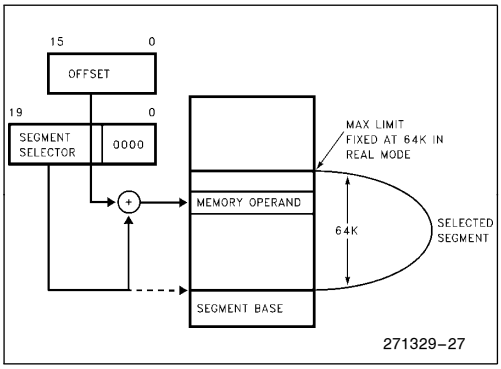


Figure 5-1. Real Address Mode Addressing

Because paging is not allowed in Real Mode, the linear addresses are the same as the physical addresses. Physical addresses are formed in Real Mode by adding the contents of the appropriate segment register, which is shifted left by four bits to an effective address. This addition results in a physi-

cal address from 00000000H to 0010FFEFH. This is compatible with 80286 Real Mode. Because segment registers are shifted left by 4 bits, Real Mode segments always start on 16-byte boundaries.

All segments in Real Mode are exactly 64-Kbytes long, and may be read, written, or executed. The Military Intel486 processor will generate an exception 13 if a data operand or instruction fetch occurs past the end of a segment (i.e., if an operand has an offset greater than FFFFH, for example, a word with a low byte at FFFFH and the high byte at 0000H).

Segments may be overlapped in Real Mode. Thus, if a particular segment does not use all 64 Kbytes, another segment can be overlaid on top of the unused portion of the previous segment. This allows the programmer to minimize the amount of physical memory needed for a program.

5.3 Reserved Locations

There are two fixed areas in memory which are reserved in Real address mode: system initialization area and the interrupt table area. Locations 00000H through 003FFH are reserved for interrupt vectors. Each one of the 256 possible interrupts has a 4-byte jump vector reserved for it. Locations FFFFFFF0H through FFFFFFFFH are reserved for system initialization.

5.4 Interrupts

Many of the exceptions shown in Table 4-16 and discussed in section 4.8.3, ‘‘Maskable Interrupt,’’ are not applicable to Real Mode operation, in particular exceptions 10, 11, 14, 17, which do not happen in Real Mode. Other exceptions have slightly different meanings in Real Mode; Table 5-2 identifies these exceptions.

5.5 Shutdown and Halt

The HALT instruction stops program execution and prevents the Military Intel486 processor from using the local bus until restarted. Either NMI, INTR with interrupts enabled (IF = 1), or RESET will force the Military Intel486 processor out of halt. If interrupted, the saved CS:IP will point to the next instruction after the HLT.

As in the case of protected mode, the shutdown will occur when a severe error is detected that prevents further processing. In Real Mode, shutdown can occur under two conditions, as follows:

- An interrupt or an exception occurs (exceptions 8 or 13) and the interrupt vector is larger than the Interrupt Descriptor Table (i.e., there is not an interrupt handler for the interrupt).
- A CALL, INT or PUSH instruction attempts to wrap around the stack segment when SP is not even (i.e., pushing a value on the stack when SP = 0001 resulting in a stack segment greater than FFFFH).

An NMI input can bring the processor out of shutdown if the Interrupt Descriptor Table limit is large enough to contain the NMI interrupt vector (at least 0017H) and the stack has enough room to contain the vector and flag information (i.e., SP is greater than 0005H). If these conditions are not met, the Military Intel486 processor is unable to execute the NMI and executes another shutdown cycle. In this case, the Military Intel486 processor remains in the shutdown and can only exit via the RESET input.

Table 5-2. Exceptions with Different Meanings in Real Mode (see Table 4-17)

Function	Interrupt Number	Related Instructions	Return Address Location
Interrupt table limit too small	8	INT Vector is not within table limit	Before Instruction
CS, DS, ES, FS, GS Segment overrun exception	13	Word memory reference beyond offset = FFFFH. An attempt to execute past the end of CS segment.	Before Instruction
SS Segment overrun exception	12	Stack Reference beyond offset = FFFFH	Before Instruction